Fakultät für Physik und Astronomie

Ruprecht-Karls-Universität Heidelberg

Bachelorarbeit
im Studiengang Physik,
vorgelegt von

Sebastian Lackner
geboren in Heidelberg, Deutschland

August 2012

Parameteroptimierung für Adaptive Exponential Integrate-and-Fire Neuronen

Diese Bachelorarbeit wurde von
Sebastian Lackner
ausgeführt am
Kirchhoff-Institut für Physik,
Ruprecht-Karls-Universität Heidelberg
unter der Betreuung von
Prof. Dr. Karlheinz Meier

Zusammenfassung

Im Rahmen dieser Arbeit werden verschiedene Algorithmen vorgestellt, um aus gegebenen Membranpotentialverläufen oder Spikezeiten eines Neurons die Parameter des Adaptive Exponential Integrate-and-Fire Modells zu extrahieren. Zuerst wird ein Algorithmus vorgestellt, der basierend auf Membranpotentialdaten eine lineare Regression durchführt, anschließend werden verschiedene Verbesserungsmöglichkeiten diskutiert. Dann werden zwei Spikebasierte Optimierungsverfahren (Partikelschwarm und Hillclimbing) hinsichtlich Konvergenzgeschwindigkeit und Qualität der Fits miteinander verglichen. Außerdem wurde ein neues Verfahren zur Abschätzung der Neuronparameter des zugrundeliegenden Modells entwickelt, welches die Schritte Neuronsimulation und Metrikberechnung miteinander kombiniert. Zuletzt wurden diese Algorithmen an einer Vielzahl von künstlichen und praxisnahen Szenarien getestet, um zu analysieren, welche Algorithmen für welche Anwendungsfälle am besten geeignet sind. Dabei wird jeweils diskutiert, inwieweit die Algorithmen auch für eine Kalibration einzelner Neuronen der BrainScaleS-Hardware geeignet sind.

Abstract

This thesis introduces several algorithms to determine neuron specific parameters of the adaptive exponential integrate-and-fire model given membrane and spike time data. First, an algorithm based on linear regression using membrane voltage traces is presented. Several extensions to improve the fit quality are discussed. Second, two spike-based optimization algorithms (particle-swarm and hillclimbing) are analyzed and compared with respect to their speed of convergence and their fit results. Third, we introduced a novel optimization algorithm to estimate the neuron parameters. It combines the steps of neuron simulation and metric evaluation. Finally, all algorithms are tested using a wide variety of artificial and measured data, in order to determine, which algorithm performs best for specific circumstances. Moreover, for each algorithm we discuss, if it is suited for calibrations of single neurons on the BrainScaleS-hardware.

Inhaltsverzeichnis

I. Einleitung

Schon lange ist es ein Ziel der Neurowissenschaft, die genaue Funktionsweise des Gehirnes sowie das Prinzip der Intelligenz besser zu verstehen. Diese Bemühungen reichen weit zurück in die Vergangenheit. Trotz technischer Fortschritte kann dieses Rätsel noch immer nicht als gelöst betrachtet werden. Parallel zur Hirnforschung im Makroskopischen (z.B. mittels Magnetresonanztomographie) wurden im 21. Jahrhundert andere alternative Methoden entwickelt, welche basierend auf mathematischen Beschreibungen von Neuronen und Synapsen eine Simulation größerer Netzwerke ermöglichen (z.B. das in Markram (2006) vorgestellte BlueBrain Projekt). Man versucht dabei sowohl das Membranpotential, als auch Aktionspotentiale (im Folgenden *Spikes* genannt) nachzubilden, wobei man davon ausgeht, dass die Information sowohl in der Rate der Spikes, als auch den genauen Feuerzeitpunkten kodiert ist (Dayan & Abbott, 2001).

Zur Realisierung solcher Simulationen ist zunächst ein geeignetes mathematisches Modell der Neuronen und Synapsen notwendig. Eines der bekanntesten Neuronmodelle ist das Hodgkin-Huxley-Modell (Hodgkin & Huxley, 1952), welches Ströme verschiedener Ionen separat erfasst, und zur Beschreibung eines Neurons 4 gekoppelte nichtlineare Differentialgleichungen benötigt. Komplexe und besonders detailgetreue Modelle erschweren jedoch sowohl die Berechnung als auch das Verständnis der Funktionsweise der Neuronen. Aus diesem Grund wurden stark vereinfachte und abstrahierte Modelle entwickelt, unter anderem das *Integrate-and-Fire*-Modell (Abbott, 1999) basierend auf nur einer linearen Differentialgleichung und 3 neuronspezifischen Parametern. Bei diesem Modell gehen allerdings ein Großteil der in der Biologie beobachteten Feuermuster von Neuronen verloren.

Einen Kompromiss bietet das *Adaptive Exponential Integrate-and-Fire*-Modell (Brette & Gerstner, 2005), im Nachfolgenden mit *AdEx* abgekürzt, welches eine Erweiterung des oben genannten *Integrate-and-Fire*-Modells darstellt. Dieses Modell basiert auf 2 gekoppelten nichtlinearen Differentialgleichungen sowie 11 neuronspezifischen Parametern, ist aber dennoch in der Lage eine Vielzahl von Feuermustern zu reproduzieren (Naud et al., 2008).

Im oben genannten BlueBrain Projekt simuliert man neuronale Netzwerke mit Supercomputern, wobei die Simulationsgeschwindigkeit durch die Detailgenauigkeit des Neuronmodells und die zur Verfügung stehende Rechenleitung begrenzt ist. Ein weiteres Problem ist der große Energieverbrauch von Supercomputern. Das BrainScaleS Projekt (BrainScaleS, 2011) (ehemals: FACETS[1]) wählt einen grundlegend anderen Ansatz, und versucht diese Einschränkungen durch die Entwicklung einer neuromorphen Hardware zu lösen, die es ermöglichen soll, auch die Simulation großer neuronalen Netzwerke bei geringem Energieverbrauch durchzuführen. Dies wird dadurch ermöglicht, dass die dem Modell zugrundeliegenden Differentialgleichungen (hier das AdEx-Modell) nicht numerisch gelöst werden - stattdessen kommen analoge

[1] Fast Analog Computing with Emergent Transient States

Schaltkreise zum Einsatz, die so konstruiert sind, dass die zeitliche Entwicklung der Spannung genau der gesuchten Lösung entspricht.

Trotz des mathematisch relativ einfachen AdEx-Neuronmodells ist es nach wie vor ein nichttriviales Problem, wenn für gegebene (z.B. biologische) Messdaten die zugrundeliegenden Parameter bestimmt werden sollen (Jolivet et al., 2008b), im nachfolgenden *Fit* genannt. Ein ähnliches Problem stellt sich auch bei neuromorpher Hardware, deren Schaltkreise kalibriert werden müssen, bevor eine Übersetzung des mathematischen Modells auf die Hardware möglich ist. Letzteres ist vor allem dann besonders wichtig, wenn die Funktionalität des Modells von möglichst exakt aufeinander abgestimmten Neuronparametern abhängt.

Im Rahmen dieser Arbeit werden wir uns damit befassen, welche Möglichkeiten und Algorithmen existieren, um eine automatische Bestimmung aller Neuronparameter des AdEx-Modells durchzuführen. Dabei werden vor allem die folgenden zwei Optimierungsszenarien betrachtet:

1. *Basierend auf dem Membranpotential*: Unter der Annahme, dass das Membranpotential eines Neurons auch ausreichend genau durch das AdEx-Modell beschrieben werden kann, kann mithilfe von vorgegebenen Membranspannungen und Spikezeitpunkten eine Anpassung durchgeführt werden. Der hierzu vorgestellte Algorithmus wurde so konzipiert, dass auch die auf der BrainScaleS Hardware vorhandenen Beschränkungen des Parameterbereichs mitberücksichtigt werden können.

2. *Basierend auf Spikezeiten*: Da Informationen durch die Feuermuster der Neuronen kodiert werden können, betrachtet man Neuronen als abstrakte Informationsverarbeitungseinheiten. In diesem Szenario soll eine Anpassung ausschließlich anhand der Spikezeitpunkte durchgeführt werden. Diese Variante ist besonders für die BrainScaleS-Hardware geeignet, da bei dieser nur für eine begrenzte Anzahl von Neuronen das Membranpotential gleichzeitig ausgelesen werden kann.

Je nachdem, um welches Optimierungsszenario es sich handelt, müssen andere Algorithmen verwendet werden, welche in der nachfolgenden Arbeit erläutert und miteinander verglichen werden. Ebenso wird untersucht, ob auch die verwendeten Referenzdaten einen Einfluss auf die Qualität der Optimierung haben.

II. Theoretische Grundlagen

In den nachfolgenden Abschnitten wird zunächst das verwendete AdEx-Neuronmodell sowie das Synapsenmodell beschrieben, deren Parameter bei gegebenen Referenzdaten automatisiert mittels Fitalgorithmen ermittelt werden sollen. Um die Algorithmen zu testen werden außerdem die verwendeten Methoden zur Erzeugung geeigneter Referenzdatensätzen sowie die theoretischen Grundlagen zum Vergleichen von Fitergebnissen erläutert.

II.1. Das AdEx-Neuronmodell

Bei dem *Adaptive exponential integrate-and-fire*-Modell (Brette & Gerstner, 2005) handelt es sich um ein single-compartement Neuronmodell, welches sich vollständig durch zwei gekoppelte, nichtlineare Differentialgleichungen sowie eine Reset-Bedingung beschreiben lässt. Diese Gleichungen beschreiben das zeitabhängige Membranpotential $V(t)$ sowie einen Adaptionsterm $w(t)$ in Abhängigkeit des injizierten externen Stroms $I(t)$. Die Reset-Bedingung dient dazu, bei einem ausreichend hohen Membranpotential ein Spike zu triggern. Die Differentialgleichungen sowie die Reset-Bedingung lauten im einzelnen:

$$\begin{aligned} C\frac{\mathrm{d}V}{\mathrm{d}t}(t) &= -g_{\mathrm{L}}\left(V(t)-E_{\mathrm{L}}\right) + g_L\Delta_{\mathrm{th}}\exp\left(\frac{V(t)-V_{\mathrm{th}}}{\Delta_{\mathrm{th}}}\right) + I(t) - w(t) \\ \tau_w\frac{\mathrm{d}w}{\mathrm{d}t}(t) &= a\left(V(t)-E_{\mathrm{L}}\right) - w(t) \\ V(t) > V_{\mathrm{peak}} &\Rightarrow V(t+\Delta t) := V_{\mathrm{reset}}, w(t+\Delta t) := w(t) + b \\ &\quad \forall\, \Delta t \in (0, t_{\mathrm{ref}}]\,,\ \text{Simulation fortsetzen bei } t + t_{\mathrm{ref}} \end{aligned} \tag{II-1}$$

Die Bedeutung der 11 Neuronparameter $C, g_{\mathrm{L}}, E_{\mathrm{L}}, V_{\mathrm{th}}, \Delta_{\mathrm{th}}, \tau_w, a, b, t_{\mathrm{ref}}, V_{\mathrm{peak}}, V_{\mathrm{reset}}$ ist Tabelle II-1 zu entnehmen. Diese Parameter werden im Nachfolgenden als zeitlich konstant angenommen. Beispiele welche basierend auf diesem Modell erzeugt wurden, sind in Abbildung II-1 bis Abbildung II-5 zu sehen.

Durch die Nichtlinearität der Differentialgleichungen sowie der Reset-Bedingung existieren keine Methoden, um das zeitliche Verhalten analytisch exakt zu berechnen, wie es z.B. ohne den Exponentialterm der Fall wäre (siehe Gleichung VII-2). Weitere Vereinfachungen sind jedoch nicht möglich, ohne dass dadurch die Vielfalt an Spike-Feuermustern eingeschränkt wird, und deutliche Abweichungen zum biologischen Vorbild auftreten (Naud et al., 2008): Durch Vernachlässigung des Exponentialterms findet kein starker Anstieg des Membranpotentials unmittelbar vor einem Spike mehr statt, und ohne die zweite Gleichung für $w(t)$ wäre das Modell nicht mehr in der Lage, durch Spikes verursachte Adaptionseffekte zu beschreiben.

Für den in Kapitel III vorgestellten Algorithmus bietet es sich an, die nachfolgenden Substitution durchzuführen, um das Gleichungssystem zu separieren:

$$w(t) := a\left(\Omega(t) - E_{\mathrm{L}}\right) + b\Theta(t) \tag{II-2}$$

Tabelle II-1.: Bedeutung der verschiedenen AdEx-Modellparameter (Brette & Gerstner, 2005)

Variable und Bedeutung		**Variable und Bedeutung**	
C	Membrankapazität	a	Unterschwellige Adaption
g_L	Leckkonduktanz	b	Durch Spikes ausgelöste Adaption
E_L	Ruhepotential	t_ref	Refraktärdauer
V_th	Schwellenspannung des Exponentialterms	V_peak	Spike Schwellspannung
Δ_th	Steigungsfaktor des Exponentialterms	V_reset	Spike Rückstellspannung
τ_w	Adaptions-Zeitkonstante		

Durch die Wahl einer geeigneten Differentialgleichung für $\Theta(t)$ erhält man die nachfolgenden, modifizierten AdEx-Differentialgleichungen sowie eine modifzierte Reset-Bedingung:

$$\begin{aligned} C\frac{\mathrm{d}V}{\mathrm{d}t}(t) &= -g_\mathrm{L}\left(V(t)-E_\mathrm{L}\right) - a\left(\Omega(t)-E_\mathrm{L}\right) \\ &\quad -b\Theta(t) + g_\mathrm{L}\Delta_\mathrm{th}\exp\left(\tfrac{V(t)-V_\mathrm{th}}{\Delta_\mathrm{th}}\right) + I(t) \\ \tau_w\frac{\mathrm{d}\Theta}{\mathrm{d}t}(t) &= -\Theta(t) \\ \tau_w\frac{\mathrm{d}\Omega}{\mathrm{d}t}(t) &= V(t)-\Omega(t) \\ V(t) > V_\mathrm{peak} &\Rightarrow V(t+\Delta t) := V_\mathrm{reset}, \Theta(t+\Delta t) := \Theta(t)+1 \\ &\quad \forall\, \Delta t \in (0, t_\mathrm{ref}]\,,\ \text{Simulation fortsetzen bei } t+t_\mathrm{ref} \end{aligned} \tag{II-3}$$

Aus physikalischer Sicht sind diese Gleichungen identisch zu Gleichung II-1, dennoch weist dieses Gleichungssystem den entscheidenden Vorteil auf, dass $\Theta(t)$ nun dimensionslos ist, und nur noch von den Spike-Zeitpunkten abhängt. Somit kann $\Theta(t)$ bei bekannten Spike-Zeitpunkten bereits im Voraus berechnet werden. Diese Vorgehensweise ist auch in Pfeil (2011) beschrieben, und wurde dort ebenfalls für die Anpassung mittels linearer Regression verwendet.

Bei der Verwendung der hierbei angegebenen Algorithmen bleibt vor allem noch zu beachten, dass sich je nach verwendetem Simulator die Neuronparameter in ihren Einheiten unterscheiden. Einige Simulatoren fordern außerdem die Angabe von $\tau_\mathrm{m} = \frac{C}{g_\mathrm{L}}$ statt g_L. Soweit nicht anderst angegeben, wurde in dieser Arbeit jeweils die pyNN-Notation verwendet (Davison et al., 2008).

II.2. Synapsenmodell

Bisher haben wir angenommen, dass dem Neuron ein beliebiger externer Strom $I(t)$ injiziert wird. Dieser Strom entspricht der Anregung des Neurons durch die Spikes andere Neuronen,

welche über eine Synapse mit diesem verbunden sind. Daher wird das zuvor erläuterte Modell zusätzlich um eine mathematische Beschreibung der Synapsen erweitert. Bei einer Anregung durch j stimulierende (*excitatory*) und k hemmende (*inhibitory*) Synapsen erhält man für den eingehenden Strom (Jolivet et al., 2008b):

$$I(t) = \sum_j g_{\mathrm{e}}(t) \cdot (V(t) - E_{\mathrm{e}}) + \sum_k g_{\mathrm{i}}(t) \cdot (V(t) - E_{\mathrm{i}}) \tag{II-4}$$

Dabei sind $g_{\mathrm{e}}(t)$ und $g_{\mathrm{i}}(t)$ anregende bzw. hemmende Konduktanzen, welche der nachfolgenden Differentialgleichung sowie Spike-Bedingung genügen ($\mathrm{k} \in \{\mathrm{e}, \mathrm{i}\}$):

$$\begin{aligned} \tau_{\mathrm{k}} \tfrac{\mathrm{d}g_{\mathrm{k}}}{\mathrm{d}t}(t) &= -g_{\mathrm{k}}(t) \\ \text{eintreffender Spike} \quad \Rightarrow \quad g_{\mathrm{k}}(t + \mathrm{d}t) &= g_{\mathrm{k}}(t) + \mu_{\mathrm{k}} \end{aligned} \tag{II-5}$$

In dieser Arbeit werden wir im Folgenden nur den vereinfachten Fall betrachten, dass lediglich *eine* äußere Anregung stattfindet. In diesem Spezialfall gibt es nur noch 3 unbekannte Parameter, mit denen sich das vollständige Verhalten beschreiben lässt. Die freien Parameter sind im Einzelnen:

$$E_{\mathrm{k}}, \tau_{\mathrm{k}}, \mu_{\mathrm{k}} \tag{II-6}$$

Dabei ist μ_{k} die Veränderung der Konduktanz bei einem eintreffenden Spike, E_k ist das Umkehrpotential, und τ_{k} die Zeitkonstante der Synapse. Durch Vereinfachen der Gleichung II-4 und Gleichung II-5 erhält man den folgenden Strom:

$$I(t) = \mu_{\mathrm{k}} \left(\sum_{\mathrm{l}} \Theta\left(t - t_l\right) \exp\left(-\frac{t_l}{\tau_{\mathrm{k}}}\right) \right) \cdot (V(t) - E_{\mathrm{k}}) \tag{II-7}$$

Dabei ist $\Theta(x)$ die Heaviside-Funktion und t_l sind die Zeitpunkte der eintreffenden Spikes.

II.3. Metriken

Um die Qualität eines Fits zu bewerten, auch wenn die Neuronparameter der Referenzdaten nicht bekannt sind, benötigt man ein Verfahren, welches einen direkten Vergleich zwischen einem Referenzdatensatz (z.B. Membranpotential und/oder Spikes) und einem simulierten Datensatz durchführt. Dazu bietet es sich an Metriken zu definieren, welche den Begriff Übereinstimmung genauer definieren, und diese somit numerisch vergleichbar machen. Für eine Metrik fordern wir mindestens die folgenden Voraussetzungen, welche ebenfalls in Born (2012) zum Einsatz kamen:

$$\begin{aligned} d : (x, y) \in \mathrm{P}^2 \quad &\rightarrow \quad \mathbb{R} \\ \forall x \in \mathrm{P} \quad &: \quad d(x, x) = 1 \\ \forall x \neq y \quad &: \quad d(x, y) \leq 1 \end{aligned} \tag{II-8}$$

Dabei ist P der Raum der möglichen Spike-Zeitpunkte, der zeitlichen Entwicklung des Membranpotentials oder eine Kombination beider Daten. Metriken spielen, abgesehen von der Beurteilung der Qualität eines Fits noch eine weitere, entscheidende Rolle: Sie eignen sich, sofern gewisse Bedingungen erfüllt sind, in Kombination mit universellen Optimierungsverfahren (wie

z.B. einem Hillclimbing-Algorithmus) ebenfalls zum Ermitteln der Fitparameter. In diesem Fall ist es besonders wichtig, dass das von der Metrik erzeugte Potential stetig ist und möglichst nur ein einziges globales Maximum besitzt (Born, 2012). In den nachfolgenden Untersuchungen kommen die folgenden Metriken zum Einsatz:

- *Spannungs-Metrik*: Bei bekannter Referenzspannung kann mit dem *root-mean-square*-Verfahren die Abweichung zwischen Referenzspannung und simulierter Spannung berechnet werden. Dieser Wert wurde über die Gleichung $1 - \exp\left(-\frac{1}{\text{RMS}}\right)$ auf den Bereich zwischen 0 und 1 beschränkt. Ein Wert von 1 entspricht hierbei einer perfekten Übereinstimmung des Membranpotentials.

- *Spike-Metrik*: Analog zu einem Spannungsvergleich ist auch ein Vergleich der Spike-Zeiten möglich. Hierbei haben wir uns für die in Victor & Purpura (1996) vorgestellte Metrik (im Nachfolgenden auch *Viktor-Purpura*-Metrik genannt) entschieden, welche mit dem in Born (2012) erläuterten Verfahren auf den Bereich zwischen 0 und 1 normiert wurde. Ein Wert von 1 entspricht auch hier einer perfekten Übereinstimmung der Spikezeitpunkte.

Zusätzlich zu diesen Metriken kann bei Referenzdaten mit bekannten Neuronenparametern ein direkter Vergleich dieser zur Beurteilung der Übereinstimmung verwendet werden. Bei Verwendung von Fitalgorithmen mit Fehlerabschätzung könnte geprüft werden, ob die vom Algorithmus ermittelten Neuronparameter innerhalb der 3σ - Umgebung der Referenzparametern liegen. Bei Fitroutinen ohne Fehlerabschätzung kann mithilfe der prozentualen Abweichung beurteilt werden, wie gut die Werte übereinstimmen. Die prozentuale Abweichung lässt sich aus den fehlerbehafteten Parametern $x_1, \ldots, x_n$ sowie den Abweichungen zu den Referenzparametern $\Delta x_1, \ldots, \Delta x_n$ berechnen mit:

$$\text{rel. Fehler} = \sqrt{\left(\frac{\Delta x_1}{x_1}\right)^2 + \ldots + \left(\frac{\Delta x_n}{x_n}\right)^2} \tag{II-9}$$

II.4. Erzeugung der Referenzdaten

Zum Vergleich verschiedener Fitalgorithmen bietet es sich an, künstlich erzeugte Referenzdaten zu verwenden, deren Neuronparameter somit bereits bekannt sind. Zur Berechnung des Membranpotentials sowie der Spikes kamen die Neuronparameter aus Tabelle II-2 (entnommen aus dem Artikel von Naud et al. (2008)) in Kombination mit verschiedenen künstlich erzeugten Stromstimuli zum Einsatz. Die dabei angebenen Werte I_{multiply} wurden als Skalierungsfaktoren verwendet, um die generierten Stromstimuli (mit Mittelwert ≈ 0.41 nA) auf die gewünschte Größenordnung zu bringen, sodass das Neuron eine übliche Spike-Ereignisrate aufweist. Diese Faktoren wurden ebenfalls in Analogie zu den Untersuchungen in Naud et al. (2008) gewählt. Die Simulationen wurden jeweils mit dem NEST-Simulator (Gewaltig & Diesmann, 2007) und einer Schrittweite von 0.1 ms durchgeführt.

Für die Wahl des Stromstimulus gibt es viele verschiedene Möglichkeiten, welche je nach verwendetem Fitalgorithmus zu unterschiedlichen Ergebnissen führen können. Zur exakten

Tabelle II-2.: In den Simulationen verwendete Neuronparameter (aus (Naud et al., 2008))

Variable		**tonic spiking (0)**	**adaption (1)**	**initial burst (2)**	**regular bursting (3)**
C	[nF]	0.2	0.2	0.13	0.2
τ_{m}	[ms]	20.0	16.667	7.222	20.0
E_{L}	[mV]	-70.0	-70.0	-58.0	-58.0
V_{th}	[mV]	-50.0	-50.0	-50.0	-50.0
Δ_{th}	[mV]	2.0	2.0	2.0	2.0
τ_{w}	[ms]	30.0	300.0	150.0	120.0
a	[nS]	2.0	2.0	4.0	2.0
b	[nA]	0.0	0.06	0.12	0.1
t_{ref}	[ms]	0.0	0.0	0.0	0.0
V_{peak}	[mV]	0.0	0.0	0.0	0.0
V_{reset}	[mV]	-58.0	-58.0	-50.0	-46.0
I_{multiply}		1.22	1.22	0.98	0.51

Bestimmung aller Parameter erscheint es sinnvoll, den Stromstimulus so zu wählen, dass sämtliche Antworten eines Neurons auf äußere Reize getestet werden, andernfalls könnten mehrere näherungsweise gleich gute Lösungen mit unterschiedlichen Neuronparametern existieren. Im Rahmen dieser Arbeit wurde eine Vielzahl von möglichen Strömen als Eingabe getestet, und somit experimentell bestimmt, welcher Stimulus bei welchem Algorithmus zu den besten Ergebnissen führt. Die nachfolgend erläuterten Stromstimuli wurden jeweils so gewählt, dass Mittelwert und Standardabweichung näherungsweise übereinstimmen, und somit eine dadurch verursachte Abweichung ausgeschlossen werden kann.

1. *Einfache Stufen*: Hierbei wird systematisch die Reaktion des Neurons auf verschiedene Stromstärken untersucht - nicht jedoch z.B. der Einfluss auf Stromschwankungen. Ähnliche Stimuli wurden häufig in biologischen Experimenten verwendet (Jolivet et al., 2008b) (siehe Abbildung II-1).

2. *Zufällige Stufen*: Um zu testen, ob der direkte Wechsel zwischen verschiedenen Stromstärken zu einer besseren Fit-Qualität führt, wurde ein weiterer Stromstimulus erzeugt, bei dem die Stufen zufällig verteilt sind (siehe Abbildung II-2). Für die Auswertung bzgl. der linearen Regression kamen zwei verschiedene mit dieser Methode erzeugte Stromstimuli zum Einsatz.

3. *Normalverteilter Stromstimulus*: Unter Verwendung eines konstanten Mittelwertes μ_{I} und einer festgelegten Standardabweichung σ_{I} wurde ein Stromstimulus mit einer Normal-

verteilung erzeugt. Dazu wurde für jeden Zeitschritt unabhängig voneinander ein zufälliger Wert gezogen (siehe Abbildung II-3). Die verwendetem Parameter zur Erzeugung beider Stromstimuli sind:

$$(\mu_\mathrm{I}, \sigma_\mathrm{I}) \in \{(0.402, 0.142), (0.428, 0.285)\} \tag{II-10}$$

4. *Postsynaptischer Stromstimulus*: Diese Vorgehensweise ist inspiriert von Kobayashi et al. (2009). Um einen synaptischen Stromstimulus zu simulieren (ähnlich der Anregung von tausenden anregenden und hemmenden postsynaptischen Strömen) erzeugt man einen Strom basierend auf folgendem mathematischen Zusammenhang:

$$\begin{aligned} I(t) &= A \sum_l I_\mathrm{exc} \cdot g_{\tau_\mathrm{exc}}(t - t_l) + A \sum_m I_\mathrm{inh} \cdot g_{\tau_\mathrm{inh}}(t - t_m) \\ g_\tau(t) &= \Theta(t) \cdot \tfrac{t}{\tau} \exp\left(-\tfrac{t}{\tau}\right) \end{aligned} \tag{II-11}$$

   Hierbei ist $\Theta(x)$ die Heaviside-Funktion, $A = 1.0$ ein Skalierungsfaktor und $I_\mathrm{exc} = 0.1$ nA, $I_\mathrm{inh} = -0.033$ nA, $\tau_\mathrm{exc} = 1$ ms sowie $\tau_\mathrm{inh} = 3$ ms Konstanten. Die Spike-Zeiten t_l, t_k wurden mithilfe eines Poissonprozesses bei vorgegebenen Raten $r_\mathrm{exc}, r_\mathrm{inh}$ erzeugt. Der multiplikative Term $\frac{t}{\tau}$ dient zur Modellierung des Anstiegs der Konduktanz. Ein Beispiel für einen solchen Strom ist in Abbildung II-4 zu sehen. Verwendete Parameter für die Erzeugung der beiden Stromstimuli:

$$(r_\mathrm{exc}, r_\mathrm{inh}) \in \{(6880.0, 2880.0), (24520.0, 20520.0)\} \tag{II-12}$$

5. *Ornstein-Uhlenbeck Stromstimulus*: Basierend auf der Vorgehensweise von Jolivet et al. (2008a) wurde ein Datensatz basierend auf dem Ornstein-Uhlenbeck-Prozess erzeugt. Dies wird durch die nachfolgende stochastische Differentialgleichung sichergestellt[1]:

$$I(t + \mathrm{d}t) := I(t) \cdot \left(1 - \tfrac{\mathrm{d}t}{\tau_\mathrm{I}}\right) + \tfrac{\mu_\mathrm{I}}{\tau_\mathrm{I}} \cdot \mathrm{d}t + \sqrt{\tfrac{2\sigma_\mathrm{I}}{\tau_\mathrm{I}}} \cdot \xi(t)\sqrt{\mathrm{d}t} \tag{II-13}$$

   Hierbei sind μ_I der Mittelwert und σ_I die gewünschte Standardabweichung des Stromstimulus, $\tau_\mathrm{I} = 1.0$ms ist die verwendete Zeitkonstante, und $\xi(t)$ sind zufällige Werte aus einer Normalverteilung mit Mittelwert 0 und einer Varianz von 1. Ein Beispiel für einen mit dieser Methode erzeugten Strom ist in Abbildung II-5 zu sehen. Die verwendeten Parameter zur Erzeugung der beiden Stromstimuli sind analog zu oben:

$$(\mu_\mathrm{I}, \sigma_\mathrm{I}) \in \{(0.402, 0.142), (0.428, 0.285)\} \tag{II-14}$$

[1] Die Gleichungen wurden bereits umgeformt, sodass direkt der Mittelwert sowie die Standardabweichung angegeben werden kann

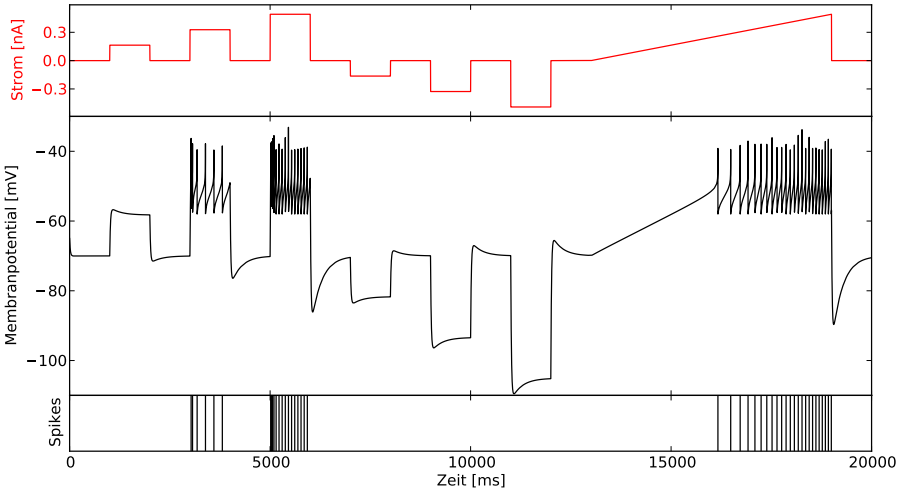


Abbildung II-1.: Stromstimulus *Einfache Stufen* (——) und mit NEST (Gewaltig & Diesmann, 2007) simuliertes Membranpotential sowie Spikes (——) unter Verwendung der Neuronparameter `adaption (1)` aus Tabelle II-2. Die Feuerrate betrug bei diesem Stimulus im Mittel 2.30 Hz.

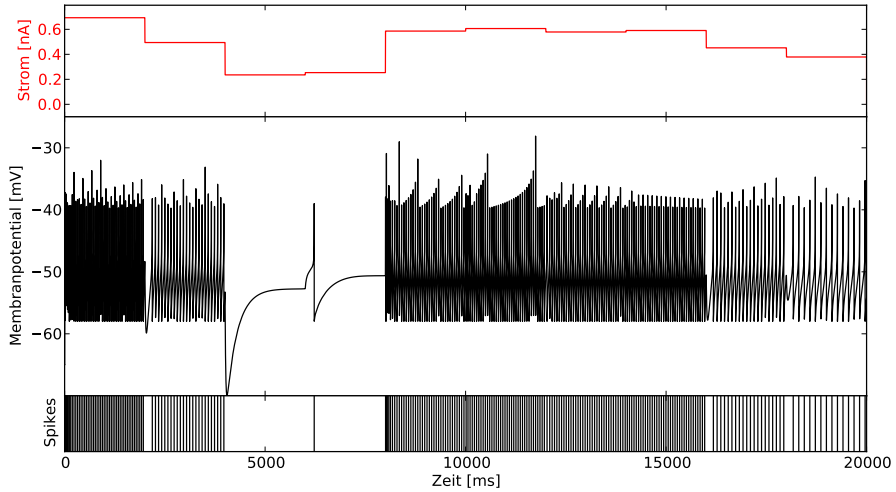


Abbildung II-2.: Beispiel für einen der beiden Stromstimuli *Zufällige Stufen* (——) und simuliertes Membranpotential sowie Spikes (——) unter Verwendung der Neuronparameter `adaption (1)` aus Tabelle II-2. Die Feuerrate betrug bei diesem Stimulus im Mittel 12.75 Hz.

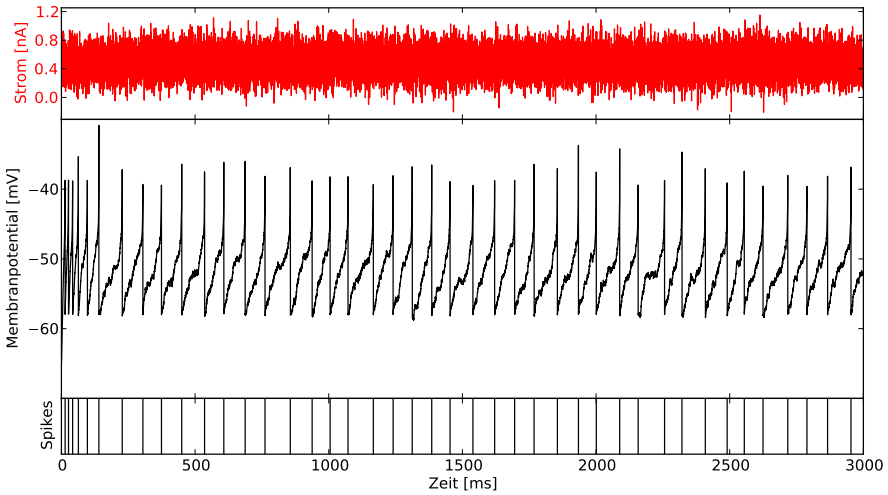


Abbildung II-3.: Ausschnitt der ersten 3 Sekunden für einen der beiden Stromstimuli *Normalverteilter Stromstimulus* (——) und simuliertes Membranpotential sowie Spikes (——) unter Verwendung der Neuronparameter `adaption (1)` aus Tabelle II-2. Die Feuerrate betrug bei diesem Stimulus 12.95 Hz.

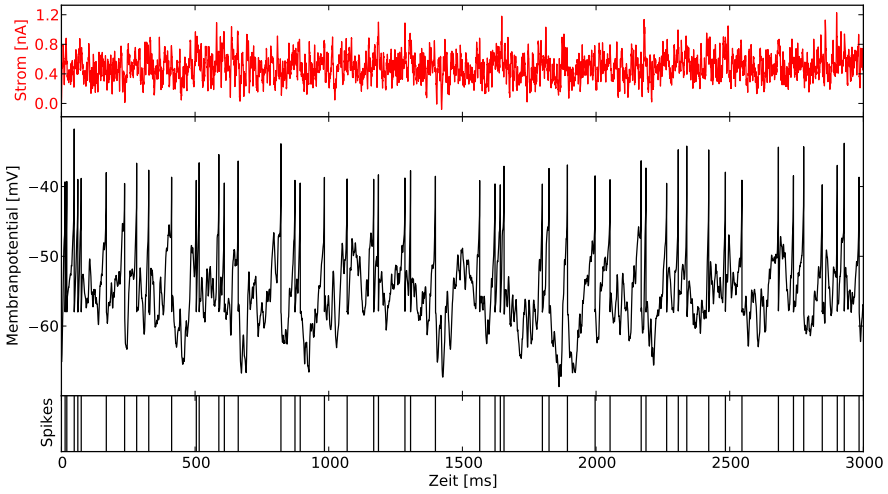


Abbildung II-4.: Aussschnitt der ersten 3 Sekunden für einen der beiden Stromstimuli *Postsynaptischer Stromstimulus* (——) und simuliertes Membranpotential sowie Spikes (——) unter Verwendung der Neuronparameter `adaption (1)` aus Tabelle II-2. Die Feuerrate betrug bei diesem Stimulus 14.70 Hz.

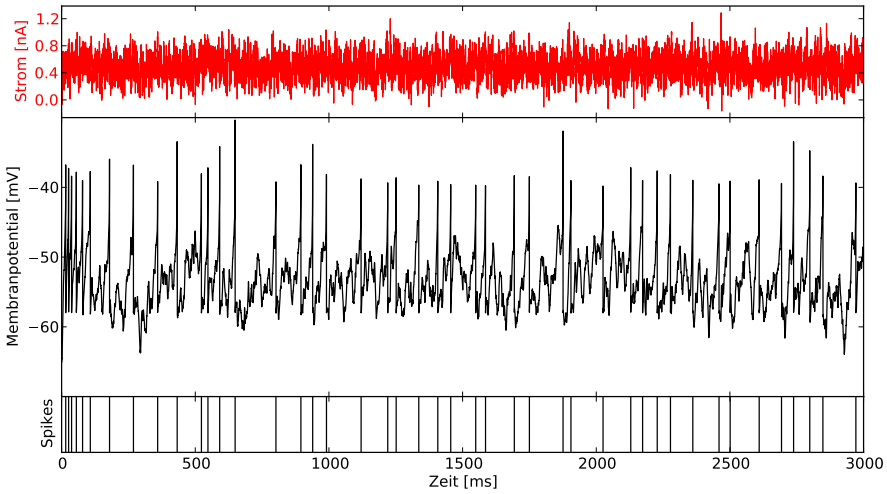


Abbildung II-5.: Aussschnitt der ersten 3 Sekunden für einen der beiden Stromstimuli *Ornstein-Uhlenbeck Stromstimulus* (——) und simuliertes Membranpotential sowie Spikes (——) unter Verwendung der Neuronparameter `adaption (1)` aus Tabelle II-2. Die Feuerrate betrug bei diesem Stimulus 13.60 Hz.

III. Potentialbasierte Anpassung von Neuronparametern

Eine Anpassung von Neuronparametern unter Verwendung des Membranpotentials ist vor allem dann wichtig, wenn man an ein anderes theoretisches Modell oder biologische Membranmessungen fitten möchte.

Die Verwendung generischer Optimierungsverfahren bringt immer einige Nachteile mit sich. Aufgrund des großen Parameterraums ist bei solchen Algorithmen eine enorme Anzahl von Simulationen notwendig und durch die Nichtlinearität der Differentialgleichungen existieren keine schnellen Lösungsverfahren. Hinzu kommt das Problem, dass bei solchen Verfahren keine Konvergenz gegen das globale Maximum garantiert werden kann.

Im Folgenden werden wir ein Anpassungsverfahren speziell für das AdEx-Neuronmodell basierend auf einer linearen Regression vorstellen, das von den obigen Problemen nicht betroffen ist. Desweiteren werden auch verschiedene Modifikationen des Verfahrens vorgestellt. In der anschließenden Auswertung wurden die Verfahren mithilfe der in Abschnitt II.4 erzeugten Referenzdaten getestet.

III.1. Lineare Regression

Abgesehen von dem Exponentialterm sind die Differentialgleichungen zur Beschreibung der Membranspannung vollständig linear - es liegt also nahe zu überprüfen, inwieweit man eine lineare Regression trotz des Exponentialterms erfolgreich anwenden kann.

Die Parameter $V_{\text{peak}}, V_{\text{reset}}, t_{\text{ref}}$ lassen sich relativ einfach direkt aus der Membranspannung extrahieren, und müssen somit nicht mehr in die nachfolgenden Optimierungsroutinen mit einbezogen werden. Die Spike Schwellspannung V_{peak} muss nicht sehr genau gewählt werden - es genügt für die nachfolgenden Untersuchungen einen Wert von 0 mV anzunehmen. Die Reset-Spannung dagegen kann z.B. anhand der Spannungsminima unmittelbar nach den Spikes abgeschätzt werden. Die Zeit t_{ref} entspricht dann der Zeitdifferenz zwischen dem Auslösen des Spikes, und dem Zeitpunkt, sobald die Spannung dieses Minimum erreicht hat (siehe auch Mensi et al. (2011)). Sofern die Messdaten eine ausreichende Anzahl an Spikes enthalten, können mithilfe dieser Methoden schon 3 der insgesamt 11 Parametern ermittelt werden.

Um die restlichen Parameter mit einer linearen Regression zu ermitteln, wurde in Pfeil (2011) aus der Membranspannung ein geeigneter Bereich mit $V(t) < V_{\text{th}}$ extrahiert, in dem der Exponentialterm näherungsweise vernachlässigt werden kann. Basierend auf Gleichung (II-3) kann zunächst $\Theta(t)$ sowie $\Omega(t)$ berechnet werden. Hierzu sind lediglich die Spike-Zeitpunkte, das Membranpotential $V(t)$, eine Schätzung von τ_{w}, sowie geeignete Anfangsbedingungen für die jeweiligen Differentialgleichungen notwendig. Anschließend kann in der vereinfachten Differentialgleichung die Ableitung $\frac{\mathrm{d}V}{\mathrm{d}t}(t)$ mithilfe von Sekanten approximiert werden. Man erhält

somit die folgende Gleichung:

$$C\frac{V(t+\mathrm{d}t)-V(t)}{\mathrm{d}t} = -g_\mathrm{L}\left(V(t)-E_\mathrm{L}\right) - a\left(\Omega(t)-E_\mathrm{L}\right) - b\Theta(t) + I(t) \tag{III-1}$$

Aus dieser Formel lässt sich das nachfolgende äquivalente Matrix-Gleichungssystem konstruieren:

$$\begin{pmatrix} V(t_0) & \Omega(t_0) & \Theta(t_0) & I(t_0) & 1 \\ V(t_1) & \Omega(t_1) & \Theta(t_1) & I(t_1) & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix} \cdot \begin{pmatrix} -\frac{g_\mathrm{L}}{C} \\ -\frac{a}{C} \\ -\frac{b}{C} \\ \frac{1}{C} \\ \frac{E_\mathrm{L}}{C}\cdot(g_\mathrm{L}+a) \end{pmatrix} = \begin{pmatrix} \frac{V(t_1)-V(t_0)}{t_1-t_0} \\ \frac{V(t_2)-V(t_1)}{t_2-t_1} \\ \vdots \end{pmatrix} \tag{III-2}$$

Zur Lösung eines solchen Gleichungssystems kann z.B. ein Least-Square-Verfahren verwendet werden. Um die vorherige Schätzung von τ_w zu verbessern kann man anschließend diesen Parameter so variieren, dass das Residuum der Gleichung minimiert wird.

Mit dieser Vorgehensweise lassen sich insgesamt 9 der 11 Parameter relativ exakt ermitteln. Wenn V_th jedoch nicht bekannt ist, dann ist unklar, welcher Bereich des Membranpotentitals für einen Fit verwendet werden kann. Außerdem ist eine lineare Regression alleine nie ausreichend, denn es besteht keine Möglichkeit auch noch die letzten beiden Parameter $V_\mathrm{th}, \Delta_\mathrm{th}$ exakt zu bestimmen.

Der nachfolgende Lösungsvorschlag basiert auf einem in Mensi et al. (2011) vorgestellten Verfahren: Man ersetzt die Exponentialfunktion durch eine Approximation mittels einer abschnittsweise konstanten Funktion.

$$\Delta_\mathrm{th}\exp\left(\tfrac{V(t)-V_\mathrm{th}}{\Delta_\mathrm{th}}\right) \approx B_i, \quad i = \left\lfloor N\cdot\tfrac{V-V_\mathrm{min}}{V_\mathrm{max}-V_\mathrm{min}}\right\rfloor \tag{III-3}$$

Dabei sind B_i die Werte der Exponentialfunktion, welche im Nachfolgenden auch mit *Bins* bezeichnet werden, V_min sowie V_max definieren den durch die Approximation abgedeckten Potentialbereich, und N entspricht der Anzahl der Bins und somit der Genauigkeit der Approximation. Dieses Vorgehen wird durch Abbildung III-1 noch einmal verdeutlicht. Dank dieser Vereinfachung kann die Exponentialfunktion nun direkt in einem linearen Gleichungssystem mitberücksichtigt werden, indem man die Höhen der Bins als freie Parameter einfügt. Dadurch wird die ursprüngliche Optimierung der 5 Parametern $g_\mathrm{L}, a, b, C, E_\mathrm{L}$ zu einer Optimierung von $5+N$ Parametern, wobei N der Anzahl der Bins und somit der Genauigkeit der Approximation der Exponentialfunktion entspricht. Um die in den N neuen Parametern enthaltene Information auszuwerten, kann anschließend mithilfe von wenigen Datenpunkten ein nichtlinearer Fit des Terms $\Delta_\mathrm{th}\exp\left(\frac{V(t)-V_\mathrm{th}}{\Delta_\mathrm{th}}\right)$ an diese durchgeführt werden. In unserer Implementierung wurde das in *SciPy* (Jones et al., 01) integrierte *downhill simplex*-Verfahren verwendet (Nelder & Mead, 1965).

Im Rahmen dieser Arbeit wurde der oben beschriebene Algorithmus implementiert, und anhand einiger Datensätze getestet. Dabei zeigte sich, dass aufgrund der Vielzahl an neu hinzugefügten Parametern, nicht mehr sichergestellt ist, dass die freien Parameter tatsächlich eine

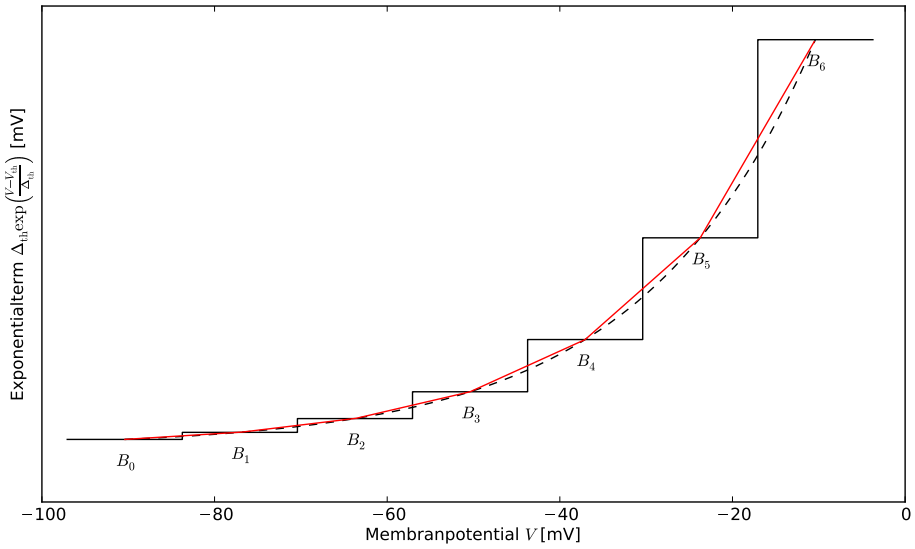


Abbildung III-1.: Veranschaulichung der im Nachfolgenden verwendeten Approximation der Exponentialfunktion mittels konstanter diskreter Bins B_i (——) im Vergleich zur exakten Formel (----). Dieser Zusammenhang wird auch in Gleichung III-3 genauer erläutert. Zusätzlich ist in dieser Abbildung die in Unterabschnitt III.1.4 vorgestellte Modifikation eingezeichnet, bei der die Exponentialfunktion stattdessen linear interpoliert wird (——).

Exponentialfunktion bilden. Dies zeigt sich dadurch, dass das Verfahren zwar in gewissen Spezialfällen durchaus sehr exakt funktioniert, aber in der Mehrzahl der Fälle (z.B. bei biologischen Messdaten) keine brauchbaren Ergebnisse mehr liefert.

Zusätzlich werden bisher noch einige Idealisierungen angenommen, die bei realen Messdaten ebenfalls nicht zwangsläufig erfüllt sein müssen (z.B. wird der zur Berücksichtigung von Adaptionseffekten notwendige Term $\Omega(t = 0)$ nicht mitgefittet und stattdessen als 0 angenommen).

Im Folgenden werden wir einige Verbesserung dieser linearen Regression vorstellen, und anschließend die Auswirkungen dieser Verbesserungen untersuchen. Die Verbesserungen beziehen sich hierbei insbesondere auf den ersten Teil des Verfahrens zur Ermittlung der N Bin-Parameter B_i, aus denen anschließend die übrigen 2 Parameter extrahiert werden können.

III.1.1. Berücksichtigung des unbekannten Anfangszustandes

Wie bereits zuvor erwähnt, wurde bei der ursprünglichen Implementierung $\Omega(t = 0)$ als konstant 0 angenommen. Dieser Term dient dazu die Adapation bedingt durch zuvor ausgelöste Spikes zu beschreiben - die Methode eignet sich deshalb nicht für biologische Messdaten mit unbekanntem Anfangswert sowie für Simulationsdaten, aus denen ein beliebiger Ausschnitt

extrahiert wurde.

Angenommen $\Omega_1(t)$ und $\Omega_2(t)$ sind Lösungen der Differentialgleichungen

$$\tau_w \frac{\mathrm{d}\Omega_1}{\mathrm{d}t}(t) = V(t) - \Omega_1(t), \quad \tau_w \frac{\mathrm{d}\Omega_2}{\mathrm{d}t}(t) = -\Omega_2(t), \tag{III-4}$$

dann gilt für eine Superposition der Form $\Omega(t) = \Omega_1(t) + \alpha\Omega_2(t)$:

$$\begin{aligned} & \tau_w \frac{\mathrm{d}\Omega}{\mathrm{d}t}(t) \\ = \; & \tau_w \frac{\mathrm{d}\Omega_1}{\mathrm{d}t}(t) + \tau_w \alpha \frac{\mathrm{d}\Omega_2}{\mathrm{d}t}(t) \\ = \; & V(t) - (\Omega_1(t) + \alpha\Omega_2(t)) \\ = \; & V(t) - \Omega(t) \end{aligned} \tag{III-5}$$

Andererseits gilt $\Omega(0) = \Omega_1(0) + \alpha\Omega_2(0)$. Aus diesen Umformungen wird ersichtlich, dass man offensichtlich im Falle $\Omega_2(0) \neq 0$ jede beliebige Lösung für $\Omega(t)$ als eine Superposition der beiden Funktionen $\Omega_1(t)$ und $\Omega_2(t)$ darstellen kann.

Diese Tatsache lässt sich ausnutzen, um $\Omega(0)$ in der Fitroutine mitzuberücksichtigen. Unter der Annahme, dass $\Omega_2(0) = 1$ erhält man $\alpha = \Omega(0) - \Omega_1(0)$ und somit das nachfolgende Gleichungssystem:

$$\begin{pmatrix} V(t_0) & \Omega_1(t_0) & \Omega_2(t_0) & \Theta(t_0) & I(t_0) & 1 \\ V(t_1) & \Omega_1(t_1) & \Omega_2(t_1) & \Theta(t_1) & I(t_1) & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix} \cdot \begin{pmatrix} -\frac{g_\mathrm{L}}{C} \\ -\frac{a}{C} \\ -\frac{a}{C} \cdot \alpha \\ -\frac{b}{C} \\ \frac{1}{C} \\ \frac{E_\mathrm{L}}{C} \cdot (g_\mathrm{L} + a) \end{pmatrix} = \begin{pmatrix} \frac{V(t_1)-V(t_0)}{t_1-t_0} \\ \frac{V(t_2)-V(t_1)}{t_2-t_1} \\ \vdots \end{pmatrix}$$

(III-6)

Diese Erweiterung kam bei allen nachfolgenden Untersuchungen zum Einsatz, da immer eine Verbesserung der Ergebnisse der linearen Regression beobachtet wurde.

III.1.2. Beschränkung der Neuronparameter

Wie zuvor bereits erwähnt, führt die große Anzahl an zusätzlichen Parametern in vielen Fällen zu einer Konvergenz gegen eine falsche Lösung. Grund ist die durch die Binparametern bedingte große Anzahl an (scheinbar) zusätzlichen Freiheitsgraden. Da die Bins B_i einer Exponentialfunktion entsprechen müssen, sind diese Parameter nicht tatsächlich frei, es ist jedoch nicht möglich die Abhängigkeiten mittels linearer Gleichungen zu formulieren, was für eine Integration in das Gleichungssystem notwendig wäre. Um dieses Problem zu minimieren, bietet es sich an, den Parameterbereich auf plausible Lösungen einzuschränken. Dies lässt sich durch das Hinzufügen einer Beschränkung der Fitparameter auf einen mehrdimensionalen Quader

realisieren.

$$\begin{pmatrix} x_{1,\min} \\ x_{2,\min} \\ x_{3,\min} \\ x_{4,\min} \\ x_{5,\min} \end{pmatrix} \leq \begin{pmatrix} -\frac{g_L}{C} \\ -\frac{a}{C} \\ -\frac{b}{C} \\ \frac{1}{C} \\ \frac{E_L}{C} \cdot (g_L + a) \end{pmatrix} \leq \begin{pmatrix} x_{1,\max} \\ x_{2,\max} \\ x_{3,\max} \\ x_{4,\max} \\ x_{5,\max} \end{pmatrix} \tag{III-7}$$

Die Ungleichungen sind dabei Komponentenweise zu verstehen. Die Werte $x_{i,\min}, x_{i,\max}$ sollten dabei so gewählt sein, dass sämtliche erlaubten Kombinationen der Werte $g_\mathrm{L}, a, b, C, E_\mathrm{L}$ in dem Quader enthalten sind.

Diese Vorgehensweise hat noch einen weiteren Vorteil: Sofern die optimalen Parameter anschließend z.B. für eine Hardware-Simulation genutzt werden sollen, und diese gewisse Beschränkungen an den Parameterbereich stellt (wie z.B. $a \geq \mathrm{const}_+ > 0$ bei der BrainScaleS-Hardware), kann dies direkt beim Durchführen der linearen Regression mitberücksichtigt werden. Im Falle von mehreren guten Lösungen, wird somit immer die gewünschte zurückgegeben.

Mathematisch lässt sich eine solche Beschränkung des Parameterbereichs z.B. mit *Bounded-Variable Least-Square*-Algorithmen (BVLS) lösen. In unserer Implementierung kam der in Stark & Parker (1995) vorgestellte Algorithmus zum Einsatz, der auf einer iterativen Lösung des Gesamtproblemes mithilfe von mehreren gewöhnlichen Least-Square-Aufrufen basiert.

III.1.3. Beschränkung der Binparameter B_i

Nach wie vor ist nicht sichergestellt, dass die N zusätzlichen Binparameter B_i tatsächlich gegen eine Exponentialfunktion konvergieren, sodass V_th und Δ_th anschließend mit einem nichtlinearen Fit ermittelt werden können. Solche Fälle kommen in der Praxis durchaus häufig vor, wie auch in der nachfolgenden Auswertung noch genauer zu sehen ist. Da in einem solchen Fall sämtliche ermittelten Fitparameter signifikant von den Erwartungen abweichen, muss eine Methode gefunden werden, die die Wahrscheinlichkeit eines solchen falschen Fits verkleinert.

Durch Gleichung III-7 wurden bereits 5 der Parameterkomponenten eingegrenzt. Eine ähnliche Beschränkung kann auch für die N Parameter, welche die Exponentialfunktion beschreiben, übernommen werden. Eine optimale Lösung kann für dieses Problem unter Verwendung von linearer Regression nicht existieren, weil es unmöglich ist, nichtlineare Terme in einer linearen Regression zu berücksichtigen. Dennoch gibt es gewisse Möglichkeiten, um Bedingungen an die Binparameter zu stellen. Diese Methode wird im Nachfolgenden kurz erläutert.

Im Folgenden seien $x_1, \ldots, x_n$ die durch einen Fit zu bestimmenden Parameter (Neuronparameter sowie Binparameter). Das ursprüngliche Gleichungssystem sei:

$$\begin{pmatrix} a_{11} & \ldots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \ldots & a_{mn} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix} \tag{III-8}$$

Angenommen die zusätzlichen Bedingungen an die Parameter $x_1, \ldots, x_n$ seien Ungleichungen und gegeben durch eine Matrix C (wobei der Vergleich wieder Komponentenweise zu

verstehen ist):

$$\begin{pmatrix} y_1 \\ \vdots \\ y_k \end{pmatrix} = \begin{pmatrix} c_{11} & \dots & c_{1n} \\ \vdots & \ddots & \vdots \\ c_{k1} & \dots & c_{kn} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \geq \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \tag{III-9}$$

Um dieses Problem mit dem BVLS-Algorithmus zu lösen, kann man die Ungleichungen der temporären Variablen $y_1, \dots, y_k$ in Einschränkungen des Parameterbereichs umformen, welche anschließend an den Algorithmus übergeben werden können:

$$\begin{pmatrix} y_1 \\ \vdots \\ y_k \end{pmatrix} \geq \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \tag{III-10}$$

Das daraus resultierende (kombinierte) Gleichungssystem lautet:

$$\underbrace{\begin{pmatrix} a_{11} & \dots & a_{1n} & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mn} & 0 & \dots & 0 \\ g \cdot c_{11} & \dots & g \cdot c_{1n} & g & 0 & \\ \vdots & \ddots & \vdots & & \ddots & \\ g \cdot c_{k1} & \dots & g \cdot c_{kn} & 0 & & g \end{pmatrix}}_{\in M_{(m+k)\times(n+k)}} \cdot \begin{pmatrix} x_1 \\ \vdots \\ x_n \\ y_1 \\ \vdots \\ y_k \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_m \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{III-11}$$

Das umgeformte Problem kann anschließend analog zur ursprünglichen Gleichung III-8 mit dem BVLS-Algorithmus gelöst werden. Der Parameter g kontrolliert hierbei, wie stark die Bedingungen erfüllt sein müssen. Für $g \to \infty$ nähert sich das Ergebnis immer weiter dem theoretisch exakten Ergebnis an. Sofern der Wert g hoch gewählt wird, kann eine ausreichend genaue Näherungslösung erhalten werden.

Die oben erläuterte Methode wurde anschließend dazu verwendet, um die folgenden Bedingungen für die Binparameter $B_1, \dots, B_N$ (siehe Abbildung III-1) zu erzwingen, welche unter anderem von Exponentialfunktionen erfüllt werden:

$$\begin{aligned} B_i &\leq B_{i+1}, \\ B_i &\leq \tfrac{1}{2}(B_{i-1} + B_{i+1}). \end{aligned} \tag{III-12}$$

Die erste Bedingung entspricht dabei der Monotonie der durch die Bins beschriebenen Funktion, die zweite Bedingung entspricht der lokalen Konvexität. Diese Bedingungen charaktierisieren die Exponentialfunktion nicht eindeutig, dennoch wird damit der mögliche Parameterbereich der Lösung stark eingeschränkt.

III.1.4. Linear interpolierte Bins

Das ursprüngliche Verfahren nutzte zur Interpolation der Exponentialfunktion lediglich diskrete Bins. Wie anhand von Abbildung III-1 jedoch deutlich wird, kann der tatsächliche Wert der Exponentialfunktion teilweise deutlich über- bzw. unterschätzt werden. In einigen Fällen hebt sich diese Abweichung im Mittel auf, dies ist jedoch nur dann der Fall, wenn die Datenpunkte innerhalb des Bins gleichverteilt sind.

Dieser Effekt kann minimiert werden, indem man die Werte zwischen einzelnen Bins B_i linear interpoliert, wie es auch in Abbildung III-1 (——) zu sehen ist. Dennoch hat diese Näherung auch Nachteile, denn dadurch wird die Exponentialfunktion in jedem Bin geringfügig überschätzt. Inwieweit diese Modifikation eine Verbesserung mit sich bringt, wird in dem nachfolgenden Auswertungsabschnitt untersucht.

III.1.5. Berechnen der Ableitung des Membranpotentials per Polynom-Fit

Wie zuvor bereits erläutert, wurde in der ursprünglichen Implementierung die Ableitung direkt über den Zusammenhang

$$\frac{\mathrm{d}V}{\mathrm{d}t}(t) \approx \frac{V(t+\mathrm{d}t) - V(t)}{\mathrm{d}t} \tag{III-13}$$

approximiert. Bei einem stetigen Verlauf des Membranpotentials ist diese Approximation gerechtfertigt und liefert gute Ergebnisse. Sobald man jedoch einen rauschenden Eingangsstrom verwendet, so überträgt sich dieses Rauschen auch auf das Membranpotential.

Im Nachfolgenden wurde getestet, ob sich diese Approximation verbessern lässt, wenn man alternative Berechnungsmethoden verwendet. Dazu wurde eine Routine implementiert, die die Ableitung innerhalb eines Intervalls Δt durch das Anfitten eines Polynoms analytisch ermittelt.

Für jeden zu berechnenden Wert $\frac{\mathrm{d}V}{\mathrm{d}t}(t_i)$ wird somit ein gewisses Intervall $[t_i - \Delta t, t_i + \Delta t]$ ausgewählt, und in diesem ein Polynom der Form

$$V(t) = \alpha + \beta \cdot (t - t_i) + \gamma \cdot (t - t_i)^2 + \dots \tag{III-14}$$

an die Datenpunkte angefittet. Die Ableitung lässt sich dann berechnen mit $\frac{\mathrm{d}V}{\mathrm{d}t}(t_i) = \beta$.

Um zu gewährleisten, dass vor allem der Bereich um t_i möglichst genau durch den Fit beschrieben wird, wurden die Datenpunkte in unmittelbarer Nähe zu t_i stärker gewichtet, als weiter entfernte. Die verwendete Gewichtungsfunktion lautet:

$$g(t - t_i) = \exp\left(-\frac{t - t_i}{\Delta t}\right) \tag{III-15}$$

III.1.6. Alternative zur äquidistanten Bin-Einteilung

Das ursprüngliche Verfahren sieht vor, dass die Bins zur Interpolation der Exponentialfunktion durch eine äquidistante Unterteilung des gesamten benötigten Intervalls erzeugt werden (siehe Gleichung III-3). Da aber z.B. bei Spikes der Anstieg des Membranpotentials besonders steil ist, wird der Bereich mit $V > V_{th}$ nur verhältnismäßig wenige Datenpunkte beinhalten, unabhängig davon, wie lange die Referenzdaten tatsächlich sind. Zur Vermeidung dieses Problems

wurde eine weitere Option eingebaut, bei der die Unterteilung des Intervalls nicht äquidistant, sondern anhand der Anzahl der im Intervall enthaltenen Datenpunkte erfolgt. Hierdurch werden Bereiche welche viele Punkte beinhalten in kleinere Bins unterteilt. Dies lässt sich so begründen, dass man durch eine feinere Unterteilung mehr Informationen über die Exponentialfunktion erhält, und diese (theoretisch) genauer approximieren kann. Der in dieser Arbeit implementierte Algorithmus versucht dabei die Bins so zu verteilen, dass die Anzahl der Datenpunkte pro Bin möglichst exakt (innerhalb gewisser Toleranzgrenzen) übereinstimmt.

Um zu Untersuchen, ob diese Modifikation tatsächlich eine genauere Ermittlung der Fitparameter mit sich bringt, wurden entsprechende Vergleichsfits durchgeführt.

III.1.7. Zusätzliche Verwendung einer Metrik

Das Verfahren basiert bisher immer darauf, dass das Residuum in Abhängigkeit von τ_{w} minimiert wird. Nach der Durchführung vieler Fits fiel jedoch auf, dass in einigen Fällen der Parameter gegen 0 oder negative Werte konvergierte, obwohl die tatsächlichen Parameter wesentlich höher lagen. Dadurch dass der Algorithmus versucht die optimale Wahl von τ_{w} zu treffen und andere Lösungen mit nichtoptimalem τ_{w} einfach ignoriert, werden solche ggf. besseren Lösungen einfach übersprungen. Um dies zu verhindern wurde eine weitere Modifikation eingebaut, die zusätzlich nach jedem erhaltenen Zwischenergebnis eine Simulation ausführt, und mithilfe der *Viktor-Purpura*-Metrik die Abweichung zwischen Messdaten und Fit ermittelt. Dies darf nicht mit dem in Kapitel IV vorgestellen Verfahren verwechselt werden, bei dem sämtliche Neuronparameter ausschließlich anhand der Information der Metrik ermittelt werden. Um eine Optimierung von τ_{w} durchzuführen, wird bei dieser Modifikation statt des Residuums der nachfolgende Ausdruck minimiert:

$$\text{Residuum} \cdot (1 - \text{Metrik}\,(\text{Referenzzeiten}, \text{Simulation}(\vec{x}))) \tag{III-16}$$

Dabei beinhaltet $\vec{x}$ die in der letzten Iteration ermittelten Neuronparameter, wobei τ_{w} der letzten Schätzung entspricht. Referenzzeiten beinhaltet die Zeitpunkte der Spikes in den Referenzdaten. Die Metrik wird genau dann 1, wenn die Zeiten der Spikes exakt übereinstimmen - in diesem Fall ist der zu minimierende Ausdruck immer 0, auch wenn das Residuum ungleich 0 ist. Im Gegensatz zur unmittelbaren Optimierung mithilfe einer Metrik sind bei dieser Vorgehensweise nur wenige Auswertungen der Metrik notwendig. Der Nutzen einer solchen Modifikation wird im Nachfolgenden noch genauer analysiert.

III.2. Auswirkungen der verschiedenen Modifikationen

Ursprünglich war geplant, zunächst den besten Fitalgorithmus aus verschiedenen Kombinationen der in Unterabschnitt III.1.1 bis Unterabschnitt III.1.7 vorgestellten Modifikationen zu ermitteln, und damit einen Stromstimulus zu finden, welcher effizientes Fitten an Referenzdaten für unterschiedliche Neurontypen (siehe Tabelle II-2) erlaubt. Bei dieser Vorgehensweise zeigte sich jedoch, dass dies nicht ohne weiteres möglich ist, da der Algorithmus trotz Modifikationen nicht für jeden Stimulus gleich gut ist. Aus diesem Grund wurde mit jeder Modifikation eine Vielzahl von Referenzdaten gefittet, sodass eine statistische Auswertung der Fitergebnisse möglich ist.

In gewissen Fällen lieferten die entsprechenden Fitroutinen kein brauchbares Ergebnis (z.B. falls τ_w gegen einen negativen Wert konvergiert), oder nur einen Teil der Ergebnisse (z.B. keine Konvergenz beim Fitten des Exponentialterms). In solchen Fällen konnte kein Maß mehr berechnet werden, da mit den erhaltenen Neuronparametern keine NEST-Simulation (Gewaltig & Diesmann, 2007) möglich war.

Im Nachfolgenden kamen als Datensätze sämtliche Kombinationen der zuvor erzeugten Ströme (siehe Abschnitt II.4) mit den verschiedenen Neuronparametern (siehe Tabelle II-2) zum Einsatz, d.h. pro Modifikation wurden jeweils $9 \cdot 4 = 36$ Fits durchgeführt. Die Modifikationen wurden nacheinander zur ursprünglichen Implementierung, ohne Beschränkung der Neuronparameter und mit $N = 50$ Bins, hinzugefügt. Später wird sich herausstellen, dass die Modifikationen zur Beschränkung der Neuronparameter, die Beschränkung der Binparameter (mit Gewichtungsfaktor $g = 1000$), sowie linear interpolierte Bins die experimentell ermittelte beste Kombination darstellt. Diese Kombination wurde für die restlichen Untersuchungen als Referenz verwendet (markiert mit **Ref**).

III.2.1. Beschränkung der Neuronparameter und Binparameter

Zunächst wurde noch einmal experimentell untersucht, welchen Einfluss eine Beschränkung der Neuronparameter auf das Fitergebnis hat. Dazu wurde eine Reihe von Datensätzen einmal mit der ursprünglichen Methode (ohne Beschränkung der Parameter) gefittet, und ein zweites Mal mit Beschränkung der Neuronparameter (die verwendeten Grenzen sind Abschnitt VII.1 zu entnehmen). Bei einem dritten Durchlauf wurden zusätzliche Beschränkungen der Binparameter hinzugefügt, um die Wahrscheinlichkeit zu erhöhen, dass diese gegen eine Exponentialfunktion konvergieren. Die erhaltenenen Neuronparametern wurden anschließend mit denjenigen verglichen, die zur Erzeugung der Referenzdatensätze benutzt wurden. Zudem wurde mit einer Softwaresimulation das Membranpotential sowie Spikes berechnet, beides wurde mit den in Abschnitt II.3 vorgestellten Metriken mit der Referenz verglichten. Die Resultate des Vergleichs sind in der nachfolgenden Tabelle III-1 zu sehen. Die Tabelle enthält außerdem exemplarisch jeweils die Ergebnisse zweier Referenzdatensätze, welche die Effekte an konkreten Beispielen verdeutlichen.

Beim Vergleichen der ersten beiden Spalten fällt auf, dass die Beschränkung der Neuronparameter nicht auf jeden Datensatz einen positiven Effekt hat. Während z.B. für den Strom *Einfache Stufen* bereits ohne Beschränkung eine Lösung innerhalb des für NEST gültigen Parameterbereichs gefunden wurde, war dies bei dem *Postsynaptischen Stromstimulus* nicht der Fall. Dieser Effekt führt dazu, dass bei den Fits mit Beschränkung der durchschnittliche relative Fehler sinkt. Somit hat diese Modifikation wie gewünscht den Effekt, dass dadurch die Wahrscheinlichkeit erhöht wird, eine plausible Lösungen zu finden und stellt eine deutliche Verbesserung zur ursprünglichen Version des Algorithmus dar. Beim Vergleich der hinteren beiden Spalten fällt auf, dass durch die zusätzliche Beschränkung der Binparameter die Anzahl der Fits mit unvollständigen Neuronparametern geringfügig reduziert werden konnte. Ebenso konnte der durchschnittliche relative Fehler weiter minimiert und die Metrik-Durchschnittswerte um $150 - 300\%$ verbessert werden. Auch bei den beiden Beispielsdatensätzen konnte das Fitergebnis deutlich verbessert werden.

Tabelle III-1.: Vergleich der Ergebnisse *ohne Beschränkung* mit den Ergebnissen nach Hinzufügen der Modifikationen *Beschränkung des Neuronparameter* (NP) sowie *Beschränkung der Binparameter* (BP)

		ohne Beschränkung	**NP**	**NP+BP**
unvollständige Neuronparam.		22%	22%	19%
Durchschnitt	rel. Fehler[1]	$(189 \pm 84)\%$	$(108 \pm 46)\%$	$(69 \pm 58)\%$
	VP Maß[1]	0.05 ± 0.04	0.04 ± 0.03	0.12 ± 0.14
	RMS Maß[1]	0.13 ± 0.07	0.16 ± 0.04	0.25 ± 0.08
Einfache Stufen	rel. Fehler	46%	46%	8.2%
	VP Maß	0.03	0.03	0.06
	RMS Maß	0.17	0.17	0.39
Postsyn. Strom	rel. Fehler	108%	86%	26%
	VP Maß	–[2]	0.09	0.58
	RMS Maß	–[2]	0.11	0.33

[1] Angaben beziehen sich auf die Datensätze, bei denen die Parameter vollständig ermittelt werden konnten. Bei dem durchschnittlichen VP/RMS-Maß sind nur erfolgreiche NEST-Simulationen beinhaltet.

[2] keine Berechnung der Metrik möglich, da die NEST-Simulation fehlschlug.

Tabelle III-2.: Vergleich der Ergebnisse bei Verwendung *diskreter Bins* und *linear interpolierter Bins*, jeweils mit aktiver *Beschränkung der Neuronparameter* und *Binparameter*

		diskrete Bins	**linear interpolierte Bins (Ref)**
unvollständige Neuronparam.		19%	19%
Durchschnitt	rel. Fehler[1]	$(69 \pm 58)\%$	$(62 \pm 60)\%$
	VP Maß[1]	0.12 ± 0.14	0.12 ± 0.15
	RMS Maß[1]	0.25 ± 0.08	0.25 ± 0.08
Einfache Stufen	rel. Fehler	8.2%	2.9%
	VP Maß	0.06	0.07
	RMS Maß	0.39	0.46
Postsyn. Strom	rel. Fehler	26%	25%
	VP Maß	0.58	0.63
	RMS Maß	0.33	0.34

[1] Angaben beziehen sich auf die Datensätze, bei denen die Parameter vollständig ermittelt werden konnten. Bei dem durchschnittlichen VP/RMS-Maß sind nur erfolgreiche NEST-Simulationen beinhaltet.

III.2.2. Linear interpolierte Bins

Während bei den bisherigen Testläufen noch diskrete Bins verwendet wurden, wollen wir im Folgenden analysieren, welchen Einfluss es hat, wenn man stattdessen linear interpolierte Bins verwendet (siehe Abbildung III-1). Die Resultate sind in Tabelle III-2 zu sehen.

Während diese Modifikation im Durchschnitt nahezu gar keinen Effekt zu haben scheint, führte sie bei den beiden Referenzdatensätzen zu einer geringfügigen Verbesserung der Metriken und einer weiteren Reduktion der relativen Fehler. Da mit den hier genannten Modifikationen bisher die besten Ergebnisse erzielt werden konnten, werden wir diese Konfiguration im Nachfolgenden jeweils als Vergleichsdatensatz nutzen (mit **Ref** markiert).

Ein Vergleich zwischen den ursprünglichen Referenzdaten und den Ergebnissen der linearen Regression (mit diesen Einstellungen) ist in Abbildung III-2 sowie Abbildung III-3 zu sehen.

III.2.3. Variation des Gewichtes g zur Beschränkung der Binparameter

Wie in den theoretischen Grundlagen erläutert, kann die „Stärke“ der zusätzlichen Bedingungen mithilfe eines Gewichtungsfaktors g gesteuert werden (siehe Gleichung III-11). Hierbei stellt sich die Frage, ob es besser ist, den Wert für g möglichst hoch zu wählen, sodass die Bedingungen in jedem Falle erzwungen werden, oder den Wert gering zu wählen, sodass Abweichungen erlaubt sind.

Um dieses Verhalten vollständig zu untersuchen wäre es notwendig, den gesamten Wertebereich abzutasten, und das unter Verwendung sämtlicher Ströme, sodass zufallsbedingte Verbesserungen ausgeschlossen werden können. Da dies in der Kürze der Zeit nicht möglich war, wurde nur exemplarisch getestet, welchen Einfluss eine Veränderung von g hat. Bisher kam jeweils der Wert $g = 1000$ zum Einsatz. Bei einzelnen Tests mit höheren Werten führte dies teilweise zu numerischen Problemen beim Durchführen des Least-Square-Verfahrens. Wir haben uns deshalb darauf beschränkt zu testen, ob der Wert auch wesentlich kleiner gewählt werden kann. Die Resultate für die Wahl von $g = 1$ sind in Tabelle III-3 zu sehen.

Offensichtlich hatte die Reduktion des Gewichtes nur einen minimalen Einfluss auf das Fitergebnis, denn die Metriken sind nach wie vor nahezu identisch. Es fällt aber auf, dass die Reduktion des Gewichtes dazu führt, dass sich die Anzahl der Fits mit unvollständigen Neuronparametern wieder erhöht. Wir haben den Faktor deshalb bei dem bisherigen Wert $g = 1000$ belassen.

III.2.4. Zusätzliche Verwendung einer Metrik

Im Nachfolgenden wurde analysiert, welchen Einfluss die Verwendung einer Metrik zur Optimierung von τ_{w} hat. Die Resultate dieser Auswertung sind in der nachfolgenden Tabelle III-4 zusammengefasst. Der größte Vorteil dieser Methode ist, dass dadurch bei sämtlichen Datensätzen ein vollständiger Parametersatz erhalten werden konnte. Die relativen Fehler bzw. Metriken blieben dagegen nahezu unverändert.

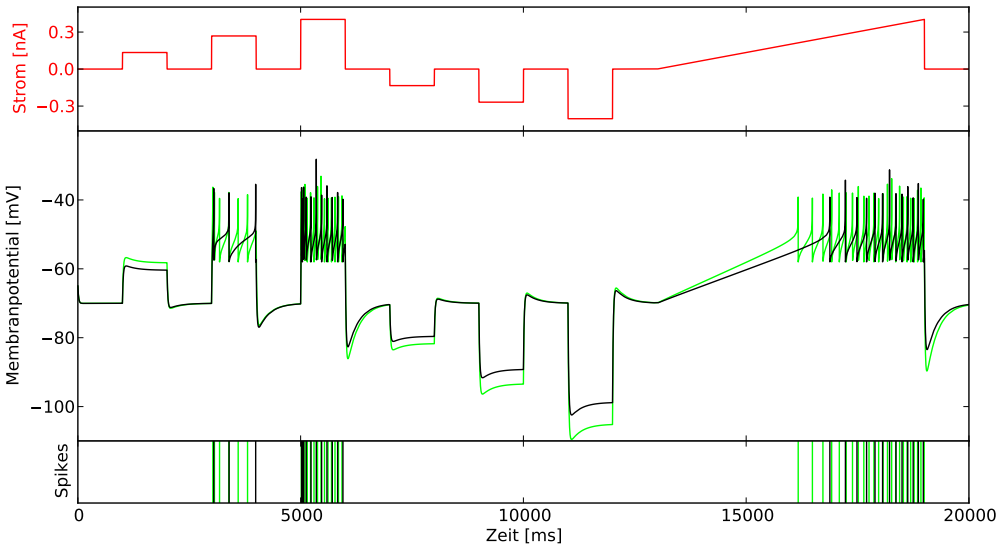


Abbildung III-2.: Vergleich der Referenz-Membranspannung und Spikes (——) mit dem Ergebnis der linearen Regression (——) beim Stromstimulus *Einfache Stufen* (——).

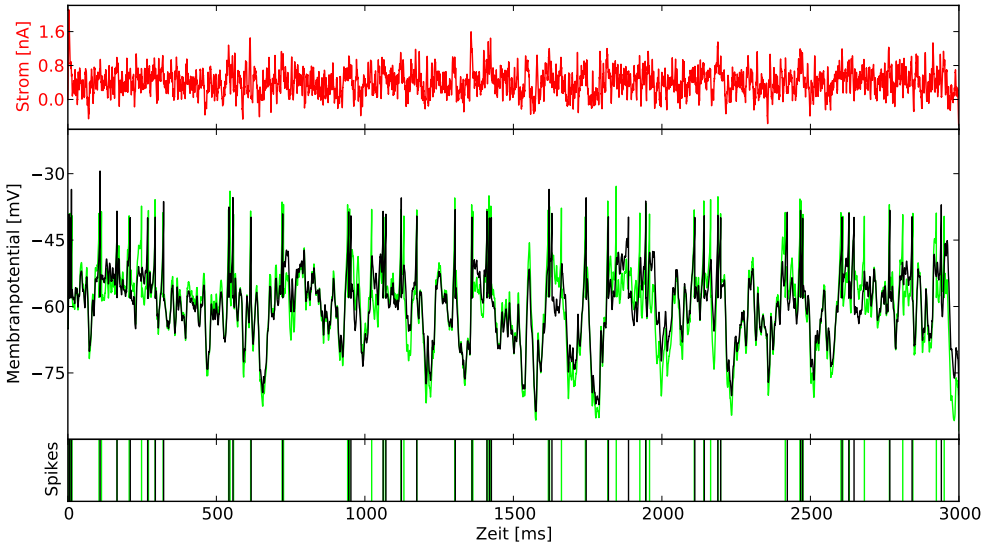


Abbildung III-3.: Vergleich der Referenz-Membranspannung und Spikes (——) mit mit dem Ergebnis der linearen Regression (——) beim Strom *Postsynaptischer Stromstimulus* (——). (Ausschnitt der ersten 3s)

Tabelle III-3.: Vergleich der Ergebnisse bei Variation des Gewichtes g zur *Beschränkung der Binparameter*. Beide Tests wurden jeweils mit aktiver *Beschränkung der Neuronparameter* durchgeführt.

		$g = 1000$ **(Ref)**	$g = 1$
unvollständige Neuronparam.		19%	22%
Durchschnitt	rel. Fehler[1]	$(62 \pm 60)\%$	$(68 \pm 63)\%$
	VP Maß[1]	0.12 ± 0.15	0.12 ± 0.16
	RMS Maß[1]	0.25 ± 0.08	0.25 ± 0.08
Einfache Stufen	rel. Fehler	2.9%	2.9%
	VP Maß	0.07	0.07
	RMS Maß	0.46	0.46
Postsyn. Strom	rel. Fehler	26%	26%
	VP Maß	0.63	0.63
	RMS Maß	0.34	0.34

[1] Angaben beziehen sich auf die Datensätze, bei denen die Parameter vollständig ermittelt werden konnten. Bei dem durchschnittlichen VP/RMS-Maß sind nur erfolgreiche NEST-Simulationen beinhaltet.

Tabelle III-4.: Vergleich der Ergebnisse beim direkten Minimieren des Residuums mit den Ergebnissen bei aktivierter Erweiterung *Mitverwendung der VP-Metrik*

		Residuum (Ref)	**Mitverwendung der VP-Metrik**
unvollständige Neuronparam.		19%	0%
Durchschnitt	rel. Fehler[1]	$(62 \pm 60)\%$	$(60 \pm 47)\%$
	VP Maß[1]	0.12 ± 0.15	0.13 ± 0.13
	RMS Maß[1]	0.25 ± 0.08	0.24 ± 0.08
Einfache Stufen	rel. Fehler	2.9%	2.8%
	VP Maß	0.07	0.05
	RMS Maß	0.46	0.37
Postsyn. Strom	rel. Fehler	26%	23%
	VP Maß	0.63	0.44
	RMS Maß	0.34	0.32

[1] Angaben beziehen sich auf die Datensätze, bei denen die Parameter vollständig ermittelt werden konnten. Bei dem durchschnittlichen VP/RMS-Maß sind nur erfolgreiche NEST-Simulationen beinhaltet.

III.2.5. Variation der Anzahl der Bins

Bei den oberen Testläufen kamen bisher jeweils $N = 50$ Bins zum Einsatz. Ob ein Erhöhen bzw. Erniedrigen der Anzahl der Bins das Ergebnis verbessert, ist in Tabelle III-5 zu sehen.

Anhand der Fitergebnisse wird deutlich, dass ein Erhöhen der Bins keine signifikante Verbesserung bringt. Der geringfügig niedrigere durchschnittliche relative Fehler steht hierbei in keinem Verhältnis zur deutlich angestiegenen Laufzeit. Auf der verwendeten Hardware[1] betrug die Laufzeit pro Datensatz bei 2 Bins gerade einmal ≈ 4 Min, bei 50 Bins bereits 5 ± 2 Std und bei 80 Bins ca. 19 ± 9 Std, jeweils abhängig davon, wie viele Iterationen notwendig waren um τ_w zu minimieren.

Exemplarisch wurde noch einmal anhand des Stromstimulus *Einfache Stufen* eine genauere Überprüfung des Einflusses der Anzahl der Bins in 10-er Schritten durchgeführt. Die Resultate sind in der nachfolgenden Abbildung III-4 visualisiert. Dabei fällt auf, dass der relative Fehler mit zunehmender Anzahl an Bins sinkt und das Viktor-Purpura-Maß steigt. Die Abbildung zeigt jedoch auch, dass bei Mitverwendung einer Metrik trotz größerem relativen Fehler ein deutlich höheres Viktor-Purpura-Ma erreicht werden konnte.

III.2.6. Berechnen der Ableitung des Membranpotentials per Polynom-Fit

Als Nächstes wurde getestet, inwiefern sich das in Unterabschnitt III.1.5 erläuterte Verfahren zur Berechnung der Steigung des Membranpotentials mittels eines Polynomfits auf das Fitergebnis auswirkt. Die Fitroutine selbst wurde hierbei so eingestellt, dass für den Fit jeweils insgesamt 9 Datenpunkte verwendet werden, und ein Polynom ersten Grades zum Einsatz kommt. Die Resultate der durchgeführten Fits sind in Tabelle III-6 zu sehen. Offensichtlich brachte diese Methode jedoch keine Verbesserung. Auf eine Untersuchung mit Polynomen höherer Grade bzw. kleineren oder größeren Ausschnitten wurde aus Zeitgründen verzichtet.

III.2.7. Alternative zur äquidistanten Bin-Einteilung

Anschließend wurde die in Unterabschnitt III.1.6 vorgschlagene Methode untersucht, die statt einer äquidistanten Einteilung des Membranpotentials eine Unterteilung wählt, sodass möglichst jeder Bin die gleiche Anzahl an Datenpunkten beinhaltet.

Auch diese Modifikation führte zu keiner Verbesserung der Fitergebnisse. Auch wenn der prozentuale Fehler in einem Fall minimal verbessert werden konnte, so ist dies wahrscheinlich auf Zufälle zurückzuführen, statt auf eine Verbesserung der Methode im allgemeinen Fall.

Die Resultate sind in Tabelle III-6 zusammengefasst.

III.2.8. Abhängigkeit der Ergebnisse vom verwendeten Stromstimulus

Entgegen der ersten Erwartungen fällt bei den Messdaten eine sehr starke Abhängigkeit vom jeweiligen Stromstimulus auf. Während manche Stromstimuli durchgehend zu sehr kleinen relativen Fehlern führten, sind die Abweichungen bei anderen Strömen deutlich größer. Dies wird auch in der nachfolgenden Abbildung III-5 noch einmal grafisch dargestellt.

[1] AMD Phenom(tm) II X4 965 Processor, 4x 3400 Mhz, der Algorithmus unterstützt bisher allerdings nur *einen* Kern

Tabelle III-5.: Auswirkungen bei Variation der Bin-Anzahl N. Alle Tests wurden jeweils mit aktiver *Beschränkung der Neuronparameter* sowie *Binparameter* durchgeführt.

		$N = 2$	$N = 50$ **(Ref)**	$N = 80$
unvollständige Neuronparam.		27%	19%	22%
Durchschnitt	rel. Fehler[1]	$(119 \pm 43)\%$	$(62 \pm 60)\%$	$(60 \pm 59)\%$
	VP Maß[1]	0.05 ± 0.05	0.12 ± 0.15	0.13 ± 0.16
	RMS Maß[1]	0.14 ± 0.05	0.25 ± 0.08	0.26 ± 0.09
Einfache Stufen	rel. Fehler	95%	2.9%	3.7%
	VP Maß	0.05	0.07	0.10
	RMS Maß	0.30	0.46	0.48
Postsyn. Strom	rel. Fehler	67%	26%	26%
	VP Maß	0.15	0.63	0.60
	RMS Maß	0.18	0.34	0.34

[1] Angaben beziehen sich auf die Datensätze, bei denen die Parameter vollständig ermittelt werden konnten. Bei dem durchschnittlichen VP/RMS-Maß sind nur erfolgreiche NEST-Simulationen beinhaltet.

Tabelle III-6.: Auswirkungen der beiden Modifikationen *Berechnen der Ableitung des Membranpotentials per Polynom-Fit* sowie *Alternative zur äquidistanten Bin-Einteilung*. Als Referenz dient die bisher besten Kombination von Modifikationen (*Beschränkung der Neuronparameter sowie Binparameter* und *linear interpolierte Bins*).

		interpolierte Bins (Ref)	**Ableitung per Polynom-Fit**	**Alt. Bin-Einteilung**
unvollständige Neuronparam.		19%	17%	19%
Durchschnitt	rel. Fehler[1]	$(62 \pm 60)\%$	$(71 \pm 61)\%$	$(63 \pm 63)\%$
	VP Maß[1]	0.12 ± 0.15	0.10 ± 0.14	0.07 ± 0.09
	RMS Maß[1]	0.25 ± 0.08	0.24 ± 0.08	0.23 ± 0.08
Einfache Stufen	rel. Fehler	2.9%	5.1%	10%
	VP Maß	0.07	0.05	0.02
	RMS Maß	0.46	0.43	0.42
Postsyn. Strom	rel. Fehler	26%	17%	32%
	VP Maß	0.63	0.68	0.21
	RMS Maß	0.34	0.40	0.27

[1] Angaben beziehen sich auf die Datensätze, bei denen die Parameter vollständig ermittelt werden konnten. Bei dem durchschnittlichen VP/RMS-Maß sind nur erfolgreiche NEST-Simulationen beinhaltet.

Desweiteren fällt auf, dass selbst bei relativ genau bestimmten Neuronparametern das Membranpotential sowie die Spikezeiten teilweise deutlich von den Referenzdaten abweichen. Dies fällt besonders stark bei Abbildung III-2 und Abbildung III-3 auf. Demnach scheint sowohl nach optischer Einschätzung, als auch nach dem Viktor-Purpura-Maß das Fitergebnis beim *Postsynaptischen Stromstimulus* (VP $= 0.63$) besser übereinzustimmen als beim Strom *Einfache Stufen* (VP $= 0.07$). Eine Ermittlung der Abweichung der Parameter liefert jedoch genau gegenteiliges, denn die Abweichung beträgt bei dem Strom *Einfache Stufen* lediglich 2.9%, während Sie beim *Postsynaptischen Stromstimulus* 26% beträgt. Daraus lässt sich schlussfolgern, dass abhängig vom verwendeten Stromstimulus die Empfindlichkeit bezüglich Abweichungen der Neuronparameter variiert.

III.2.9. Überblick über die Auswirkungen der Modifikationen

Die zuvor analysierten Unterschiede werden durch die nachfolgende Abbildung III-6 noch einmal visualisiert. Dazu wurde für 5 exemplarische Stöme der relative Fehler in Abhängigkeit des verwendeten Algorithmus aufgetragen. Hierbei ist deutlich zu sehen, dass bei den Datenpunkten *(B) beschränkter Parameterbereich*, *(C) Beschränkung der Binparameter* und *(D) interpolierte Bins* die Fehler immer weiter reduziert werden konnten. Die restlichen Modifikationen bringen nur für bestimmte Datensätze eine Verbesserung.

III.3. Zusammenfassung

In diesem Kapitel wurde ein Algorithmus zur Bestimmung der Neuronparameter aus dem Membranpotential sowie dem Stromstimulus mittels linearer Regression vorgestellt, anschließend wurden verschiedene Verbesserungsmöglichkeiten analysiert.

Während der Algorithmus in der ursprünglichen Form nur bei sehr speziellen Kombinationen von Membranpotentialverlauf und Stromstimulus ein brauchbares Ergebnis lieferte, so führen die hier vorgestellten Modifikationen dazu, dass der Algorithmus auf eine größere Klasse von Datensätzen anwendbar ist. Die Beschränkung der Neuronparameter brachte die größte Verbesserung. Durch diese Modifikation konnte in einigen Fällen, in denen die Lösung außerhalb des gewünschten Parameterbereichs lag, dennoch ein bestmögliches Ergebnis innerhalb des Parameterbereichs erhalten werden. In Fällen, in denen jedoch bereits zuvor eine Lösung innerhalb des gewünschten Parameterbereichs vorlag, brachte diese Modifikation keine Veränderungen.

Auch das Hinzufügen von Beschränkungen der Binparameter sowie die linear interpolierten Bins brachten eine Verbesserung. Die anderen vorgeschlagenen Modifikationen dagegen scheinen sich im Mittel nicht positiv auf das Fitergebnis auszuwirken. Auch wenn vereinzelt noch bessere Ergebnisse erzielt werden konnten, so sind diese Modifikationen dennoch ungeeignet, da sie bei anderen Daten das Ergebnis verschlechtern (siehe Abbildung III-6).

Bei der Auswertung fällt desweiteren auf, dass das Ergebnis sehr stark von dem jeweils verwendeten Stromstimulus abhängt. Hierbei scheinen besonders Fluktuationen im Stromstimulus und somit auch im Membranpotential zu einer Verschlechterung des Fitergebnisses zu führen. Ebenso weist die hier vorgestellte Methode den Nachteil auf, dass die Differentialgleichun-

gen lediglich lokal angepasst werden - es gibt dadurch keinerlei Kontrolle, ob auch der gesamte aus den Fitparametern resultierende Kurvenverlauf übereinstimmt.

Insgesamt hat sich die Hoffnung, dass der Algorithmus so robust ist, dass er sich auf nahezu jeden Datensatz anwenden lässt, nicht bestätigt. Schon bei simulierten Daten weichen die Ergebnisse in vielen Fällen von den Referenzdaten deutlich ab. Das Ergebnis kann verbessert werden, wenn man den erlaubten Parameterbereich weiter eingrenzt. Dies erfordert jedoch die Interaktion mit dem Benutzer, da die Parametergrenzen für jeden durchzuführenden Fit manuell anzupassen sind. Ebenso wären auch weitere Bedingungen denkbar, welche die durch die Bins verursachten zusätzlichen Freiheitsgrade einschränken, sodass der Algorithmus mit einer höheren Wahrscheinlichkeit sinnvolle Ergebnisse zurückliefert. Eine vollautomatische Methode um den Algorithmus auf Kosten einer geringeren Genauigkeit etwas robuster zu machen, stellt die Verwendung einer Metrik beim Anpassen von τ_w dar.

Mit jeder zusätzlichen Bedingung verlängert sich jedoch auch die Rechenzeit des Algorithmus, welche bei der verwendeten Hardware[2] schon bei 50 Bins ca. 5 ± 2 Std pro Datensatz betrug (je nach Anzahl der Iterationen die notwendig waren, um τ_w zu optimieren). Auch wenn es durchaus noch Optimierungsmöglichkeiten gibt, so ist es dennoch undenkbar, mit diesem Algorithmus in kurzer Zeit die Neuronparameter exakt zu bestimmen. Aus diesem Grund werden wir uns in dem nachfolgenden Kapitel mit alternativen Algorithmen beschäftigen.

Trotz der zuvor aufgezeigten Nachteile gibt es aber andere Anwendungsmöglichkeiten des Algorithmus: Unter der Voraussetzung dass eine Messung der Membranspannung zur Verfügung steht, ist es möglich, mit bereits wenigen Iterationen einen Startwert für nachfolgende Optimierungsverfahren zu finden.

[2] AMD Phenom(tm) II X4 965 Processor, 4x 3400 Mhz, der Algorithmus unterstützt bisher allerdings nur *einen* Kern

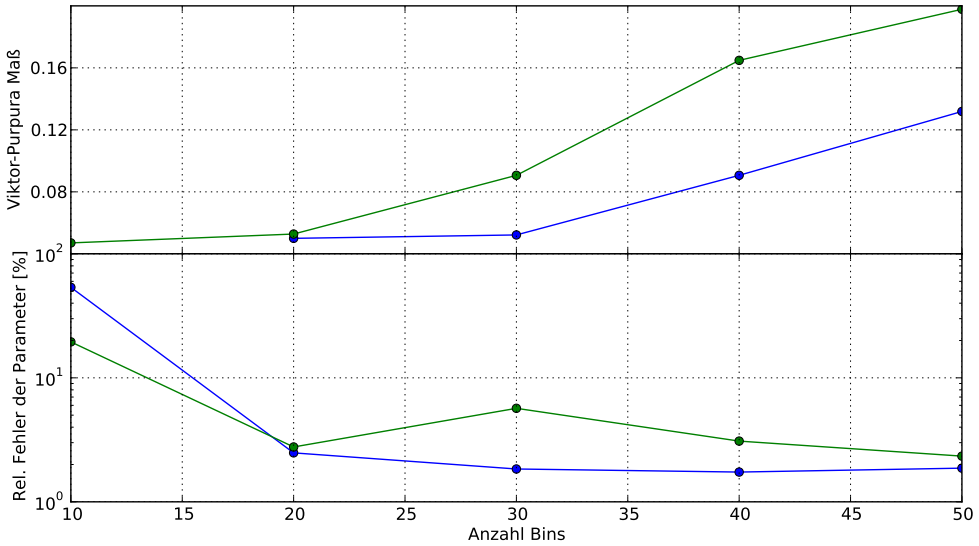


Abbildung III-4.: Vergleich der relativen Fehler der Neuronparameter und Viktor-Purpura-Metrik für die beiden Modifikationen *interpolierte Bins (Ref)* (—●—) und *Verwendung einer Metrik* (—●—) (siehe auch Unterabschnitt III.2.4), in Abhängigkeit der Anzahl der Bins zur Approximation der Exponentialfunktion. Die Fits wurden mit dem Stromstimulus *Einfache Stufen* und den Neuronparametern `adaption (1)` aus Tabelle II-2 durchgeführt. Bei $N = 10$ Bins war in einem Fall keine Simulation möglich, und somit konnte kein VP-Maß berechnet werden.

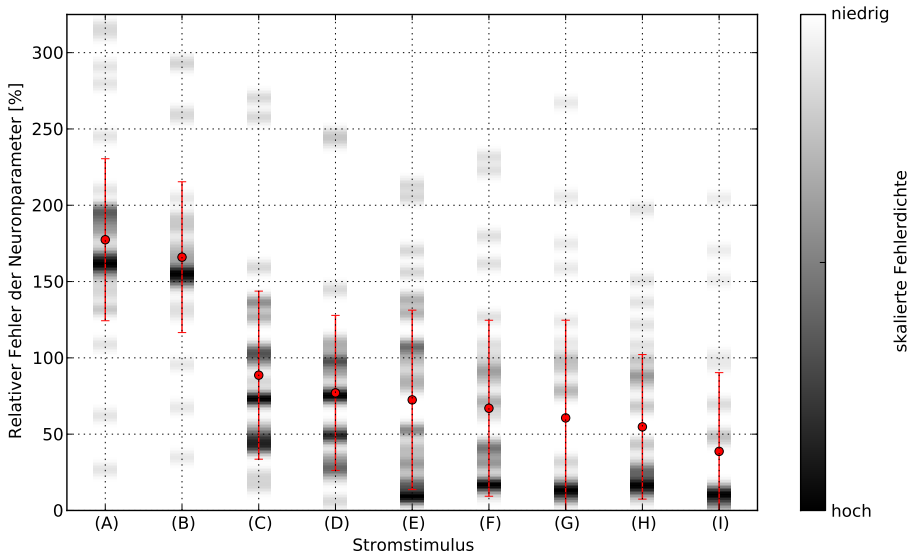


Abbildung III-5.: Relativer Fehler der Neuronparameter in Abhängigkeit des verwendeten Stromstimulus. Der Farbverlauf veranschaulicht die Häufigkeit der relativen Fehler über alle ausgewerteten Datensätze, zusätzlich wurde jeweils der mittlere Fehler (•) sowie die Standardabweichung eingezeichnet. Bei beiden *Normalverteilten Stromstimuli* (A) & (B) ist der relative Fehler der Parameter am Größten, die *Ornstein-Uhlenbeck Stromstimuli* (C) & (D) konnten den Fehler schon um ca. 50% verringern. Eine weitere Reduktion des Fehlers brachten beide *Zufällige Stufen*-Stromstimuli (E) & (G), die *Postsynaptischen Stromstimuli* (F) & (H), sowie der Strom *Einfache Stufen* (I), der im Durchschnitt zu den geringsten relativen Fehlern führte. Beide Ströme gleicher Art unterscheiden sich jeweils durch ihre Standardabweichungen (siehe Abschnitt II.4) - ein Zusammenhang dieser mit dem Fitergebnis kann jedoch nicht festgestellt werden.

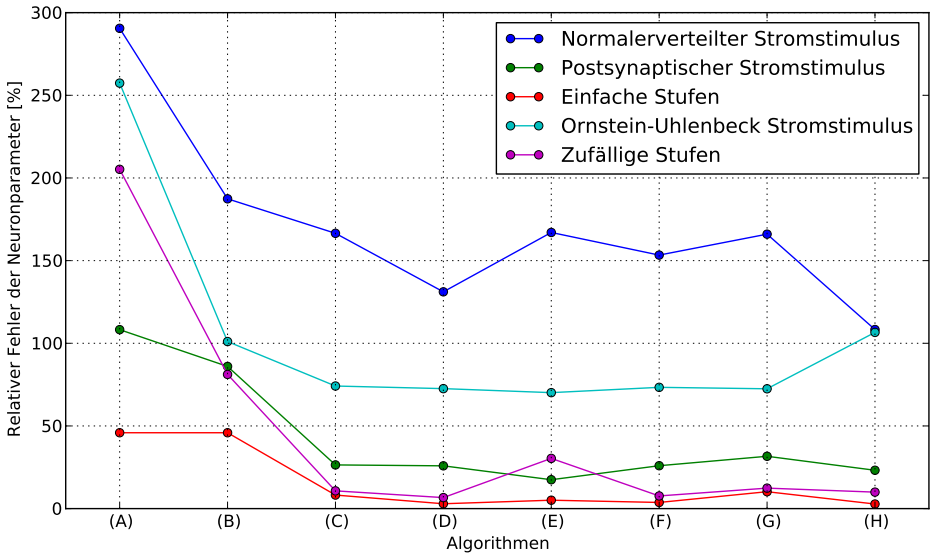


Abbildung III-6.: Relativer Fehler der Neuronparameter in Abhängigkeit der verwendeten Modifikationen. Als Referenz wurden die Parameter `adaption (1)` aus Tabelle II-2 verwendet. Getestet wurde hierbei die ursprüngliche Variante *ohne Beschränkung des Parameterebereichs* (A), die Modifikation der *Beschränkung der Neuronparameter* (B), das Hinzufügen von *Beschränkungen der Binparameter* (C) sowie das Verwenden von *linear interpolierten Bins* (D), wobei die Features jeweils nacheinander hinzugeschaltet wurden. Die nachfolgenden Modifikationen wurden jeweils einzeln hinzugeschaltet: *Berechnen der Ableitung des Membranpotentials per Polynom-Fit* (E), *Variation der Anzahl der Bins (80)* (F), *Modifikation der Bin-Einteilung* (G) sowie *Zusätzliche Verwendung einer Metrik* (H).

IV. Spikebasierte Anpassung von Neuronparametern

Das im vorherigen Kapitel vorgestellte Verfahren zur Anpassung der Neuronparameter mittels linearer Regression ist nicht in jedem Fall anwendbar. Dies liegt meist daran, dass durch die zusätzlichen Freiheitsgrade keine Sicherheit mehr besteht, dass auch tatsächlich die optimalen Neuronparameter gefunden werden. Ein weiteres Problem stellt die Analyse von Daten dar, wenn keine Messung der Membranspannung zur Verfügung steht.

Wir werden uns deshalb im Folgenden mit zwei Verfahren (Hillclimbing-Algorithmus, Partikelschwarm-Optimierung) beschäftigen, welche keine Membranspannungen benötigen. Beide Verfahren nutzen zur Optimierung eine Spike-Metrik (siehe Abschnitt II.3). Die in unserer Implementierung verwendete *Viktor-Purpura*-Metrik vergleicht die Spike-Zeitpunkte zweier Datensätze (in unserem Fall zwischen Referenzzeiten und simulierten Spikezeiten) und liefert 1 bei exakter Übereinstimmung zurück, sonst Werte kleiner als 1.

Im Rahmen dieser Arbeit wird unter anderem die Konvergenzgeschwindigkeit sowie die Effektivität beider Verfahren miteinander verglichen.

IV.1. Hillclimbing Verfahren

Im Folgenden wird zunächst die prinzipielle Vorgehensweise des Hillclimbing-Verfahrens[1] erläutert. Die Details zu verschiedenen Implementierungsmöglichkeiten können z.B. in *Global Optimization Algorithms* von Weise (2009) nachgelesen werden.

Ein Hillclimbing-Algorithmus weiß nichts über die zugrundeliegende Bedeutung der einzelnen Komponenten des Parameter-Vektors $\vec{x}$, sondern versucht mithilfe von mehreren Auswertungen das globale Maximum einer Bewertungsfunktion $S(\vec{x})$ zu finden. Dazu werden die Parameter schrittweise in beide Richtungen $\vec{x} \pm \mathrm{d}x_i \cdot \vec{e_i}$ variiert, und die Bewertungsfunktion für diese Werte berechnet ($\vec{e_i}$ sind dabei die Einheitsvektoren und $\mathrm{d}\vec{x}$ der Vektor der aktuellen Schrittweiten). Angenommen es gilt $S(\vec{x} + \mathrm{d}x_i \cdot \vec{e_i}) > S(\vec{x})$, dann befindet sich der neue Punkt näher am lokalen Maximum, und es wird die Zuweisung $\vec{x} := \vec{x} + \mathrm{d}x_i \cdot \vec{e_i}$ ausgeführt. Diese Vorgehensweise wird so lange wiederholt, bis sich der Wert von $S(\vec{x})$ in keine Richtung weiter verbessert.

Das in unserer Implementierung verwendete Hillclimbing-Verfahren arbeitet mit folgender Bewertungs-Funktion:

$$S(\vec{x}) = \text{Metrik}\left(\text{Referenzzeiten}, \text{Simulation}(\vec{x})\right) \tag{IV-1}$$

Dabei ist $\vec{x}$ ein Vektor, der sämtliche Neuron-Parameter beinhaltet, die für eine Simulation notwendig sind. Die Funktion $\text{Simulation}(\vec{x})$ liefert die aus den Parametern resultierenden Spikes

[1] In Literatur teilweise auch als „Bergsteiger-Algorithmus“ bezeichnet

zurück, welche wiederrum durch die Funktion Metrik(Referenzzeiten, ...) mit den Referenzdaten verglichen werden können.

Die oben beschriebene Implementierung hätte allerdings einige Nachteile, unter anderem, dass ein Maximum nie genauer bestimmt werden kann, als durch die Schrittweiten $\mathrm{d}x_i$ vorgegeben. Aus diesem Grund wurde der Algorithmus so modifiziert, dass die Schrittweite nach jeder Iteration angepasst wird. Zum Einen wird die Schrittweite automatisch erhöht, wenn das Maximum noch weit entfernt ist, zum Anderen werden die Schritte verkleinert, wenn man sich bereits in der Nähe des Ziels befindet.

Zur Realisierung der automatischen Schrittweitenkontrolle werden für jede Richtungskomponente x_i pro *Hillclimbing-Iteration* folgende Simulationen durchgeführt (Brownlee, 2011):

$$\begin{gathered} \vec{x} \pm d \cdot \vec{e_i} \\ d \in \left\{ -a_{\text{large}} \cdot \mathrm{d}x_i, -a_{\text{small}} \cdot \mathrm{d}x_i, a_{\text{small}} \cdot \mathrm{d}x_i, a_{\text{large}} \cdot \mathrm{d}x_i \right\} \end{gathered} \tag{IV-2}$$

Die Parameter a_{large} und a_{small} kontrollieren hierbei die Beschleunigung, und wurde in der Implementierung mit 1.2 und 0.83 initialisiert. Sollte sich die Bewertungsfunktion für die Parameter $\vec{x} \pm d \cdot \vec{e_i}$ verbessern, dann findet anschließend die Zuweisung $\mathrm{d}x_i := d$ statt. Wenn also ein großer Schritt die beste Verbesserung bringt, dann beschleunigt der Algorithmus bezüglich dieser Richtungskomponente, und wenn eine geringe Verbesserung vorteilhaft ist, dann verringert das Verfahren die Schrittweite automatisch. Falls die Variation sämtlicher Komponenten keinerlei Verbesserung mehr bringt, dann wird in unserer Implementierung die Schrittweite verkleinert.

Bisher wurde angenommen, dass bereits ein sinnvoller Startwert $\vec{x}$ vorliegt, bei dem die Suche nach dem Maximum begonnen werden kann. Bei Verwendung einer Metrik, die im Parameterraum lediglich ein globales Maximum besitzt (dieser Aspekt wurde z.B. in Born (2012) genauer untersucht), kann hierbei fast jeder beliebige Wert verwendet werden[2], und der Algorithmus wird immer zum gleichen Ergebnis kommen. Dies trifft allerdings für die von uns verwendete Bewertungsfunktion nicht zu - der Algorithmus wurde deshalb so implementiert, dass mehrere Startwerte probiert werden, um die Wahrscheinlichkeit zu erhöhen, dass das globale Maximum gefunden wird.

Die Verbesserung der Metrik mittels *Hillclimbing-Iterationen* wird so lange fortgesetzt, bis eine gewisse Genauigkeit erreicht wurde, und anschließend ein neuer Startwert generiert. In der Referenzimplementierung ist die Konvergenz erreicht, wenn sich der Wert der Metrik bei drei aufeinanderfolgenden Iterationen um weniger als 0.01% ändert. Da bei jeder Iteration die Schrittweite angepasst wird, liegt das Maximum (unter der Annahme der Stetigkeit der Metrik) anschließend in unmittelbarer Nähe der aktuellen Position $\vec{x}$.

Die Auswahl eines zu großen Startparameterbereichs, der durch alle getesteten Startwerte nicht vollständig abgedeckt wird, kann dazu führen, dass das globale Maxima nicht gefunden wird. Um bereits mit einem Teilbereich des Parameterraums zu beginnen, in dem sich wahrscheinlich die gesuchte Lösung befindet, wurde Testweise ein Verfahren zur Abschätzung der Neuronparameter anhand der Spikezeiten implementiert, welches in Abschnitt IV.3 noch genauer beschrieben wird. Der Algorithmus liefert jeweils den Mittelwert und die Standardab-

[2] Vorausgesetzt die Steigung ist ungleich 0

weichung der einzelnen Neuronparameter zurück. Mithilfe der dadurch beschriebenen Normalverteilung können anschließend Startwerte für jeden Neuronparameter gezogen werden. Auch ohne Verwendung des Abschätzungs-Algorithmus besteht die Möglichkeit die Mittelwerte/Standardabweichungen von Hand an den Hillclimbing-Algorithmus zu übergeben, welche dann auch als Normalverteilung interpretiert werden.

Alternativ besteht die Möglichkeit, für jeden Neuronparameter eine Auswahl an Möglichkeiten anzugeben. In diesem Fall ergeben sich die Startparameter als Produktraum aus beliebigen Kombinationen dieser Möglichkeiten (im Folgenden „Gitter“ genannt). Aufgrund des 11 bzw. 14-dimensionalen Parameterbereichs steigt die Anzahl der benötigten Gitterpunkte jedoch enorm an, wenn man Parameter in kleinen Schritten abtasten möchte. Die für die nachfolgenden Auswertungen verwendeten Mittelwerte/Standardabweichungen sowie Möglichkeiten für jeden Neuronparameter sind in Abschnitt VII.1 aufgelistet.

Zusätzlich wurden weitere Einstellungenmöglichkeiten implementiert, welche die Reihenfolge der Modifikation verschiedener Richtungskomponenten $\mathrm{d}x_i$ steuern.

Im sog. iterativen Richtungsmodus werden pro Hillclimbing-Iteration sämtliche Komponenten x_i des Neuronparametervektors $\vec{x}$ entsprechend der aktuellen Schrittweite $\mathrm{d}x_i$ nacheinander modifiziert. Nachdem alle 11 bzw. 14 Komponenten der Neuronparameter einmal angepasst wurden beginnt dieser Vorgang wieder von vorne. Diese Methode hat im Bezug auf parallisierter Ausführung den Nachteil, dass schon nach wenigen Simulationen wieder eine Synchronisation mit dem Hauptthread notwendig ist, der alle Ergebnisse für eine Fortsetzung der Ausführung benötigt.

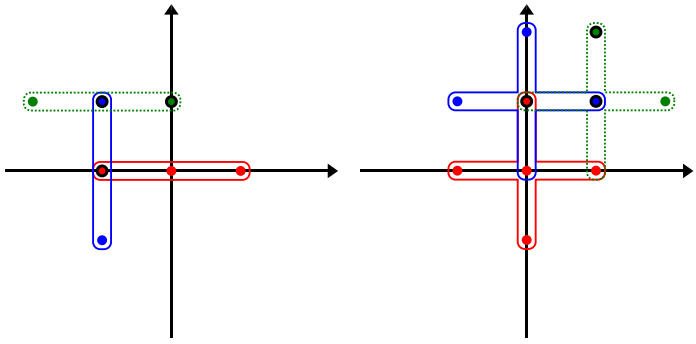
Im sog. parallelen Richtungsmodus werden in einer Hillclimbing-Iteration alle Richtungskomponenten parallel abgearbeitet, und anschließend basierend auf den ermittelten Bewertungen nur der Schritt ausgeführt, welcher die Metrik maximiert. Hierbei wird eine Synchronisation erst nach der Durchführung sämtlicher Simulationen notwendig, wodurch weniger Zeit mit Warten verbracht wird. Dieser Modus ist dadurch motiviert, dass sich in gewissen Fällen eine optimistisch gewählte Richtung als schlechte Wahl herausstellen kann, wenn man die gesamte Umgebung von $\vec{x}$ beachtet. Statt vieler (z.T. sehr kleiner Schritte) konzentriert sich dieser Modus also immer auf die Richtungskomponenten, welcher das Maß am stärksten beeinflussen. Die nachfolgende Abbildung IV-1 stellt die Unterschiede der beiden Verfahren noch einmal schematisch im 2-dimensionalen Fall dar.

Inwiefern diese Annahmen auch in der Praxis stimmen werden wir in der Auswertung noch genauer untersuchen. Ebenso wird auch untersucht, ob die Wahl des Startparameter-Verfahrens einen Einfluss auf das Fitergebnis hat. Weitere Verbesserungsideen, wie z.B. das Durchführen diagonaler Schritte, oder stochastisches Hillclimbing, bei dem die zu verbessernde Richtungskomponente zufällig ermittelt wird, wurden aus Zeitgründen zunächst vernachlässigt.

IV.2. Partikelschwarm-Optimierung

Um die Effizienz des zuvor erläuterten Hillclimbing-Algorithmus (siehe Abschnitt IV.1) zu bewerten, wird dieser mit der Partikelschwarm-Optimierung (auch PSO genannt) basierend auf Eberhart & Kennedy (1995) verglichen (siehe auch Pfeil (2011)).

Während Hillclimbing immer nur einen aktuellen Berechnungspfad verfolgt, wird bei der



(a) iterativer Richtungsmodus (b) paralleler Richtungsmodus

Abbildung IV-1.: Veranschaulichung des sog. iterativen und parallelen Richtungsmodus. Die farblichen Markierungen entsprechend den in der 1. Iteration (●), 2. Iteration (●) sowie der 3. Iteration (●) durchgeführten Simulationen im 2-dimensionalen Parameterraum, die schwarzen Umrandungen stellen die jeweils gefundenen Verbesserungsmöglichkeiten dar, welche dann den Ausgangspunkt für weitere Optimierungen bilden. Vereinfacht wurde angenommen, dass jeweils nur ein Schritt in beide Richtungen vorgenommen wird, ohne dass sich die Schrittweite ändert.

PSO eine Liste von Berechnungspfaden (Partikel genannt) parallel verwaltet. Ein weiterer Unterschied ist, dass bei der PSO der Pfad zufällig gewählt wird, Hillclimbing dagegen führt nur solche Schritte aus, die die Metrik auch tatsächlich verbessern. Zur Erzeugung des Schwarmverhaltens der Partikel wird nach jedem Rechenschritt die Geschwindigkeit und Position basierend auf dem globalen Maximum und der bisher besten Position des Partikels im Parameterraum angepasst (für Details siehe Rossant et al. (2010)).

Bezüglich der Effizienz ist hierbei anzumerken, dass diese Berechnungspfade natürlich nicht tatsächlich vollständig parallel ausgeführt werden können - die Anzahl der Simulationen welche für eine Iteration notwendig sind, hängt von der gewählten Partikelanzahl ab. Die Startparameter können auch hier entweder aus einer Normalverteilung gezogen werden werden, oder als Produktraum der Möglichkeiten für jeden Neuronparameter generiert werden.

IV.3. Verfahren zur Abschätzung der Neuronparameter

Ziel war es ein Verfahren zu konstruieren, welches eine schnelle Abschätzung der Neuronparameter unter Verwendung der Spikezeiten ermöglicht, um somit den Parameterbereich des Hillclimbing- sowie PSO-Algorithmus einzugrenzen. Dies dient zum Einen zur Reduktion der Laufzeit der Optimierungsverfahren, zum Anderen um eine Konvergenz gegen ein lokales Maximum zu vermeiden. Zur Eingrenzung des Parameterbereichs muss für jeden der Neuronparameter ein geschätzter Mittelwert sowie eine Fehlerangabe berechnet werden, was auch als Normalverteilung interpretiert werden kann.

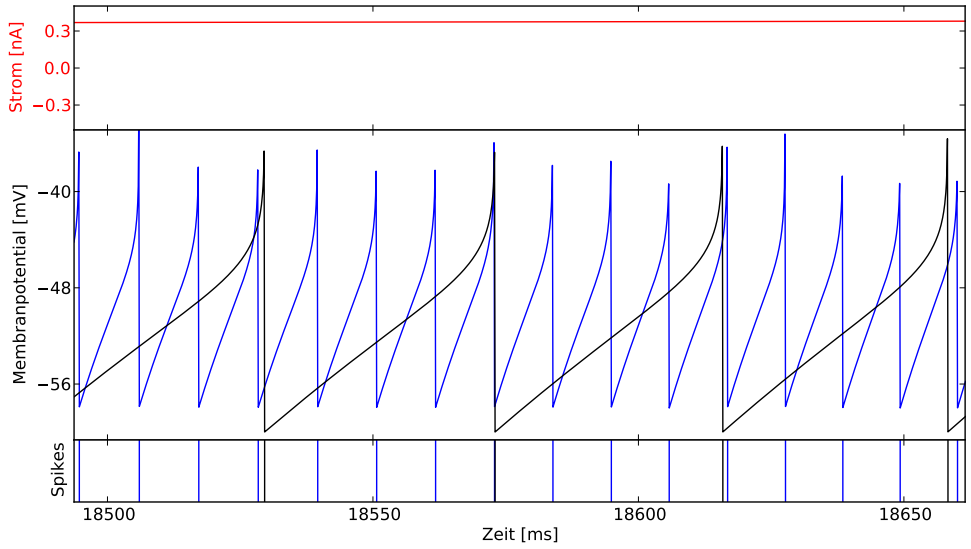


Abbildung IV-2.: Die Abbildung zeigt einen Ausschnitt eines Referenzdatensatzes (——) sowie das Ergebnis der Anpassung (——). Aufgrund der falschen Zuordnung der Spikes kann das Hillclimbing-Verfahren nicht fortgesetzt werden, da zunächst eine Verschlechterung der Metrik in Kauf genommen werden muss, um das lokale Maximum zu überwinden.

Alle bisher beschriebenen Verfahren sind für die Ermittlung der Parameter dieser Normalverteilung ungeeignet - ein Verfahren wie lineare Regression kann nicht verwendet werden, da keine Informationen über das Membranpotential vorliegen, und bei Hillclimbing sowie PSO kann aus einer besten Lösung nach wenigen Iterationen noch nicht gefolgert werden, dass das globale Maximum auch tatsächlich in der Nähe dieser Lösung liegt. Grund hierfür ist, dass die verwendete *Viktor-Purpura*-Metrik eine Zuordnung der Spikes ausschließlich anhand der Zeiten durchführt. Sobald allerdings eine gute Lösung bei falscher Zuordnung der Spikes (wie z.B. in Abbildung IV-2 zu sehen) gefunden wurde, kann das Hillclimbing-Verfahren keine weiteren Optimierungen vornehmen, ohne das Ergebnis zwischenzeitlich wieder zu verschlechtern, und scheitert deshalb.

Um dieses Problem zu lösen wurde der nachfolgende Algorithmus konzipiert, der nicht nur eine Spikemetrik darstellt, sondern auch die dazu notwendigen Simulationen selbst durchführt. Dadurch kann die Simulation dem bisherigen Ergebnis angepasst werden, und eine Fehlzuordnung der Spikes ist unwahrscheinlicher.

Die intern verwendete Simulation könnte prinzipiell mit jedem Neuronsimulator, oder auch mit der BrainScaleS-Hardware erfolgen. Da wir jedoch annehmen, dass eine Abschätzung der Neuronparameter auch unter Verwendung von Näherungslösungen der Differentialgleichungen möglich ist, wurde in dieser Arbeit ein in Python geschriebener Softwaresimulator getestet,

der zur Vereinfachung der Rechnungen den Strom zwischen zwei aufeinanderfolgenden Spikes durch den Mittelwert annähert. Die sonstigen verwendeten Näherungen werden im Anhang (Abschnitt VII.2) genauer beschrieben. Ein direkter Geschwindigkeitsvergleich mit NEST oder anderen Simulatoren ist aufgrund der verschiedenen Programmiersprachen allerdings nicht möglich.

Das Verfahren selbst besteht (ähnlich der auf Spikemetriken basierenden Optimierung) ebenfalls aus einer von den Neuronparametern abhängigen Evaluationsfunktion, welche anschließend mit einem beliebigen Optimierungsverfahren minimiert werden kann (im Gegensatz zu den hier verwendeten auf 1 normierten Metriken, welche maximiert werden müssen). In unserer Implementierung kam das in *SciPy* (Jones et al., 01) integrierte *downhill simplex*-Verfahren zum Einsatz (Nelder & Mead, 1965).

Um einen Fehler der Neuronparameter zu bestimmen wird das Verfahren mehrfach auf Ausschnitte der ursprünglichen Daten angewendet, und dann aus den resultierenden Parametern Mittelwert und Standardabweichung berechnet. In der hier verwendeten Implementierung wurden jeweils 3 Werte berechnet, und die Ausschnitte wurden durch schrittweises Entfernen einer gewissen Anzahl von Spikes am Anfang generiert.

IV.3.1. Motivation der Vorgehensweise

Analog zur hier verwendeten Viktor-Purpura-Metrik möchte man, dass die Spikezeitpunkte seiner Referenzdaten mit der Simulation möglichst exakt übereinstimmen.

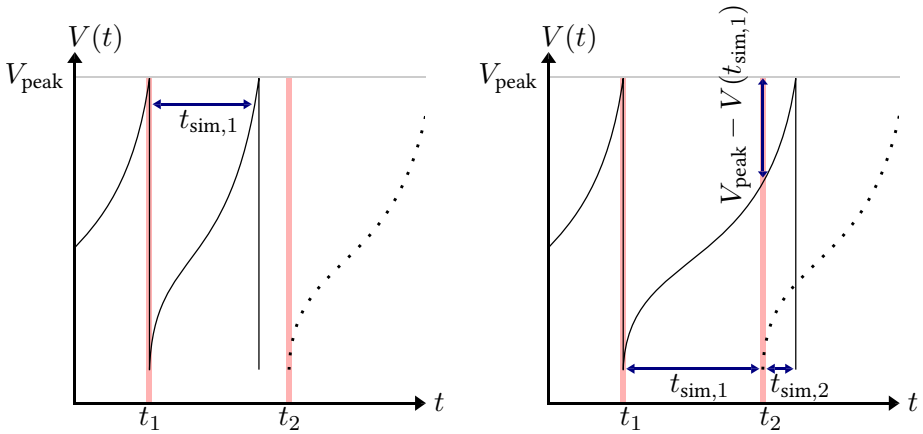
Die VP-Metrik[3] hat jedoch das Problem, dass diese keinerlei Informationen zurückliefert, wenn die Simulation keine Spikes enthält. Dies wird im nachfolgenden Algorithmus dadurch gelöst, dass in diesem Fall die Differenz des Membranpotentials, zum Zeitpunkt des erwarteten Spikes, zur Spike Schwellspannung mit in die Bewertung einfließt. Durch das Minimieren dieser Differenz erreicht man, dass das Neuron mit einer höheren Wahrscheinlichkeit feuert. Sobald das Neuron schließlich feuert, wird der Fehler durch die Verschiebung der Spikes und nicht mehr durch die Abweichung des Membranpotentials bestimmt. Da lediglich das simulierte Membranpotential verwendet wird, ist das Verfahren dennoch als spikebasierte Anpassung einzustufen. Durch die im Nachfolgenden erläuterte Vorgehensweise wird außerdem der Fall berücksichtigt, wenn ein Neuron aufgrund geringfügiger Abweichungen des Membranpotentials nicht feuert. Der dadurch fehlende Spike trägt in diesem Fall weniger stark zum Fehler bei.

Ein weiteres Problem der VP-Metrik ist, dass aufgrund der fehlenden Gewichtung der einzelnen Kosten keine iterative Anpassung einzelner Spikes erfolgt. Ein Optimierungsalgorithmus versucht die Anpassung sämtlicher Spikezeiten parallel, obwohl aufgrund der zugrundeliegenden Differentialgleichungen eine Abhängigkeit zwischen den Zeiten aufeinanderfolgender Spikes besteht. Aus diesem Grund wurde ein Gewichtungsfaktor q hinzugefügt, dernach jedem Spike angepasst wird. Dadurch wird erreicht, dass nach der signifikanten Abweichung eines Spikes die nachfolgenden Fehler kaum noch eine Rolle spielen.

[3] weitere Metriken die nach dem gleichen Prinzip arbeiten, und somit auch anfällig für die genannten Probleme sind, werden z.B. in Born (2012) diskutiert

IV.3.2. Beschreibung der Evaluationsfunktion

Der Algorithmus basiert auf einer schrittweisen Simulation des Neuronmodells zwischen aufeinanderfolgenden Spike-Zeitpunkten. Mit dem ersten Referenzspike zur Zeit t_1 beginnt die Simulation, und endet spätestens zum Zeitpunkt t_2 des nachfolgenden Referenzspikes. Im Gegensatz zu den zuvor erläuterten Ansätzen, bei denen jeweils die Gesamtdauer des Referenzdatensatzes simuliert wird, wird in diesem Fall die Simulation nach dem nächsten (in der Simulation) auftretenden Spike oder nach Ablauf der Zeit $t_{\text{max},1} = t_2 - t_1$ unterbrochen. Die tatsächliche Dauer der Simulation $t_{\text{sim},1}$ ist also immer kleiner als $t_{\text{max},1}$. Sollten die Neuronparameter bereits optimal gewählt sein, so treten die Ereignisse *Ablauf der Zeit* und *Spike* gleichzeitig auf.



(b) $t_{\text{sim},1} < t_{\text{max},1}$ (d) $t_{\text{sim},1} = t_{\text{max},1}$ und kein Spike

Abbildung IV-3.: Die Abbildung zeigt schematisch das Vorgehen zur Berechnung der Evaluationsfunktion (siehe Unterabschnitt IV.3.2) basierend auf dem simulierten Membranpotential (——) und den Positionen der Referenzspikes (——). Im linken Schaubild tritt der zum Zeitpunkt t_2 erwartete Spike bereits zu früh auf, in diesem Fall fließt in den Fehler die zeitliche Verschiebung $t_{\text{max},1} - t_{\text{sim},1}$ ein. Im rechten Schaubild fand bis zum Zeitpunkt t_2 noch kein Spike statt. Daraufhin wird die Simulation um eine gewisse Zeit $t_{\text{max},2}$ (welche von $V_{\text{peak}} - V(t_{\text{sim}})$ abhängt) fortgesetzt. Falls der Spike mit der Verzögerung $t_{\text{sim},2}$ auftritt, dann geht diese Verzögerung in die Berechnung des Fehlers ein. In beiden Fällen wird die Simulation anschließend zum Zeitpunkt t_2 fortgesetzt (·····).

Im Nachfolgenden ist die Vorgehensweise beschrieben, wie der Algorithmus weiter vorgeht, wenn die Parameter noch nicht optimal gewählt sind (andernfalls können beide nachfolgenden Schritte übersprungen werden, da diese den Fehler nicht erhöhen). Die beiden möglichen Fälle sowie die prinzipielle Vorgehensweise wird in Abbildung IV-3 visualisiert.

- Sofern noch vor Ablauf der Zeit $t_{\text{max},1}$ ein Spikeereignis stattfand, dann ergibt sich der aktuelle Fehler basierend auf der Zeitdifferenz zwischen erwartetem Spike-Zeitpunkt und

tatsächlichem Spike-Zeitpunkt:

$$\Delta E := (t_{\text{max},1} - t_{\text{sim},1})^2 \tag{IV-3}$$

Im Gegensatz zu den bisherigen Optimierungsverfahren wird anschließend die Simulation basierend auf dem zuletzt aktuellen Neuronzustand $(V(t_{\text{sim},1}), w(t_{\text{sim},1}))$ nach dem erwarteten Spike zum Zeitpunkt t_2 fortgesetzt. Das Zeitintervall $[t_1 + t_{\text{sim}}, t_2]$ und die dazugehörigen Strom bzw. Spikestimuli werden also übersprungen.

- Falls bis zum Ende der Simulationszeit noch kein Spike auftrat, so wird basierend auf dem letzten Membranpotential $V(t_{\text{sim},1})$ die Simulation um maximal $t_{\text{max},2}$ verlägert, wobei

$$t_{\text{max},2} := c_1 \cdot \left(V_{\text{peak}} - V(t_{\text{sim},1})\right)^2 + c_2. \tag{IV-4}$$

Im Folgenden wurden dazu die Konstanten $c_1 = 1\ \frac{\text{ms}}{\text{mV}^2}, c_2 = 0.1$ ms verwendet. Eine Untersuchung der Auswirkungen durch Variation von $c_1,\ c_2$ war aus zeitlichen Gründen nicht möglich. Das entscheidende an dieser Formel ist die Abhängigkeit von der Differenz zwischen Spike Schwellspannung und dem Membranpotentital, wodurch auch ohne auftretende Spikes Information mit einfließt, wie die Parameter modifiziert werden müssen. Außerdem sollte $t_{\text{max},2}$ besser überschätzt werden als unterschätzt, da andernfalls zu spät auftretende Spikes nicht mehr korrekt zugeordnet werden können.

Anschließend wird zum Zeitpunkt t_2 eine erneute Simulation gestartet, bei der unter Annahme eines gleichbleibenden Stromstimulus und einer maximalen Simulationsdauer von $t_{\text{max},2}$ überprüft wird, ob ein Spike auftritt. Ist dies nach der Zeit $t_{\text{sim},2}$ der Fall, dann gilt:

$$\Delta E := (t_{\text{sim},2})^2 \tag{IV-5}$$

Andernfalls wird einfach die maximal erlaubte Verlängerung der Simulation zur Berechnung des Fehlers verwendet:

$$\Delta E := (t_{\text{max},2})^2 \tag{IV-6}$$

Der nächste Simulationsschritt statet anschließend zum Zeitpunkt t_2.

In beiden Fällen wird der Fehler ΔE aufsummiert um den Gesamtfehler E zu erhalten. Unter Berücksichtigung des Gewichtungsfaktors des aktuellen Spikes q und der Normierung des Fehlers E_{norm} enthält man:

$$\begin{aligned} E &:= E + q \cdot \Delta E \\ E_{\text{norm}} &:= E_{\text{norm}} + q \\ q &:= q \cdot \exp\left(-\tfrac{\Delta E}{c_3}\right). \end{aligned} \tag{IV-7}$$

Diese Variablen werden zu Beginn der Funktion mit $E := 0, E_{\text{norm}} := 0$ und $q := 1$ initialisiert. Als Konstante wurde in unserer Implementierung willkürlich $c_3 = 50$ms gewählt, noch besser wäre allerdings eine von der Feuerrate abhängige Wahl, oder alternativ eine Berechnung des relativen Fehlers.

Die Simulation wird anschließend entsprechend dem obigen Verfahren so lange schrittweise fortgesetzt, bis keine weiteren Referenzspikes mehr vorhanden sind. Alternativ wäre es auch möglich bereits dann abzubrechen, wenn der Gewichtungsfaktor q so klein ist, dass sich E nicht mehr ändert. Der Rückgabewert des Algorithmus ist $\frac{E}{E_{\text{norm}}}$.

Die Normierung dient bei diesem Algorithmus nicht dazu um den Wertebereich des Fehlers einzuschränken, sondern um einen „fairen“ Vergleich zweier Ergebnisse zu ermöglichen - andernfalls wäre möglicherweise ein großer Fehler direkt bei dem ersten Spike (und somit der Vernachlässigung aller weiterer Fehler) günstiger, als eine Summe von wenigen großen Einzelfehlern.

IV.3.3. Übertragung auf die BrainScaleS-Hardware

In diesem Abschnitt wird auf die prinzipielle Übertragbarkeit dieses Algorithmus auf die BrainScaleS-Hardware eingegangen. In der obigen Fassung lässt sich der Algorithmus noch nicht direkt auf die Hardware anwenden, da die Simulation schrittweise stattfindet, und nur Teile des Stromstimulus benutzt werden bzw. konstante Bereiche hinzugefügt werden. Dennoch existieren gewisse Möglichkeiten, wie man den obigen Algorithmus in modifizierter Form implementieren könnte. Im Rahmen der Bachelorarbeit war es aus zeitlichen Gründen nicht möglich, diese Ideen weiter zu verfolgen.

Da eine schrittweise Simulation sowie das anschließende Fortsetzen mit exakt definierten Neuronparametern nicht möglich ist, muss stattdessen der Strom bzw. Spikestimulus selbst angepasst werden, und die Simulation jeweils wieder von Anfang an gestartet werden. Man führt also zunächst eine Simulation mit dem ursprünglichen Stimulus durch, und modifiziert diesen dann iterativ durch Entfernen gewisser Bereiche bzw. Hinzufügen von konstanten Bereichen[4], wenn nach dem in Unterabschnitt IV.3.2 erläuterten Verfahren ein zeitlicher Sprung stattfindet. Nach Abbildung IV-3 würde also im Falle $t_{\text{sim},1} < t_{\text{max},1}$ das Intervall $[t_1 + t_{\text{sim},1}, t_2]$ entfernt werden, sowie im Falle $t_{\text{sim},1} = t_{\text{max},1}$ zur Zeit t_2 ein konstanter Strom der Dauer $t_{\text{sim},2}$ eingefügt werden. Diese Vorgehensweise hat allerdings im Vergleich zur computerbasierten Simulation den Nachteil, dass für jeden einzelnen Referenzspike mindestens eine Simulation gestartet werden muss. Um die Anzahl der Simulationen zu reduzieren wäre es denkbar, jeweils mehrere Spikes auf einmal zu simulieren, bevor eine Anpassung des Stimulus stattfindet.

Ein weiteres Problem bei der Implementierung des Algorithmus auf der Hardware ist die fehlende Möglichkeit zum gleichzeitigen Auslesen des Membranpotentials $V(t_{\text{sim},1})$ beliebig vieler Neuronen. Um dennoch eine Rückmeldung über die Entfernung bis zur Spike Schwellspannung zu erhalten, kann alternativ der Stimulus so verändert werden (unter Verwendung eines hohen Stromes bzw. einer hohen Feuerrate), dass das Neuron innerhalb kurzer Zeit spiked. Die Zeitdifferenz bis zum nächsten Spike erlaubt anschließend Rückschlüsse auf das Membranpotential vor den angefügten Daten. Alternativ wäre auch eine Approximation mittels Softwaresimulationen möglich, da diese Werte nicht besonders exakt sein müssen, und lediglich zur Vorgabe der Optimierungsrichtung dienen.

[4] Bei Verwendung eines Spikestimulus ist stattdessen eine konstante Feuerrate denkbar

IV.4. Vergleich zwischen Hillclimbing und PSO

Zum Vergleich beider Algorithmen wurden diese für mehrere Referenzdatensätze mit bekannten Neuronparametern ausgeführt. Für eine Bewertung soll zum Einen festgestellt werden, ob das Hillclimbing-Verfahren wie gewünscht funktioniert, welcher der beiden Algorithmen (Hillclimbing oder PSO) im Durchschnitt zu besseren Ergebnissen führt, und durch welche Einstellungen ein gutes Ergebnis bzw. eine geringe Laufzeit erreicht werden kann. Außerdem soll überprüft werden, ob der in Abschnitt IV.3 vorgestellte Algorithmus zur Abschätzung der Neuronparameter zu einem signifikanten Unterschied bezüglich der Konvergenzgeschwindigkeit oder der Ergebnisse führt. Pro Algorithmus und Einstellung wurden insgesamt 10 Fits (2 verschiede Neuronparameter kombiniert mit 5 verschiedenen Strömen) durchgeführt - eine Verwendung aller Kombinationen der Parameter und Ströme aus Kapitel III war aus zeitlichen Gründen nicht möglich.

IV.4.1. Maximale Verbesserung durch Hillclimbing-Iterationen

Vor dem Vergleich beider Algorithmen miteinander wurde zunächst überprüft, ob das Hillclimbing-Verfahren überhaupt den Wert der Metrik der zufälligen Startparameter weiter verbessern kann. Dies dient hier vor allem zur Überprüfung, ob der Algorithmus wie gewünscht funktioniert. Es wäre jedoch auch denkbar eine solche Untersuchung zum Vergleich verschiedener Metriken einzusetzen, um damit die Untersuchungen aus Born (2012) fortzusetzen.

Basierend auf den ausgewerteten Datensätzen wurde zunächst ermittelt, wie viele Hillclimbing-Iterationen nötig sind, bevor die Abbruchbedingung einen nächsten Startwert probiert. Die Resultate hierzu sind in Tabelle IV-1 zu sehen.

Im Folgenden sei $S_{\text{iter}}(i)$ der Wert der Metrik nach i Hillclimbing-Iterationen. Die relative Verbesserung im i-ten Schritt ist dann gegeben durch $\frac{S_{\text{iter}}(n+1)-S_{\text{iter}}(n)}{S_{\text{iter}}(n)}$. In der nachfolgenden Abbildung IV-4 ist zu sehen, wie gut sich im Durchschnitt[5] die Viktor-Purpura-Metrik der Startparameter pro Hillclimbing-Iteration verbessert.

Um abzuschätzen, wie groß die maximale Gesamtverbesserung ist, wenn man beliebig viele Hillclimbing-Iterationen durchführen würde, wurde anhand der experimentell ermittelten Daten ein Fit mit der nachfolgenden Gleichung durchgeführt:

$$\frac{S_{\text{iter}}(n+1) - S_{\text{iter}}(n)}{S_{\text{iter}}(n)} = a \cdot b^n \tag{IV-8}$$

Diese Gleichung kann so interpretiert werden, dass pro Hillclimbing-Iteration die durchschnittliche relative Verbesserung um den Faktor b abnimmt, und für $n = 0$ genau a beträgt. Dieser Zusammenhang wurde auch als Fit in die dazugehörige Abbildung eingezeichnet, die Fitkurven stimmen dabei für beide Datensätze nahezu exakt überein.

Unter der Annahme dass Gleichung IV-8 auch für beliebig große n im Mittel erfüllt ist, lässt sich nun die maximale prozentuale Gesamtverbesserung abschätzen mit:

$$\frac{S_{\text{iter}}(\infty)}{S_{\text{iter}}(1)} = \prod_{i=1}^{\infty} \frac{S_{\text{iter}}(i+1)}{S_{\text{iter}}(i)} = \prod_{i=1}^{\infty} \left(1 + a \cdot b^i\right) \tag{IV-9}$$

[5] gemittelt über die verschiedenen Datensätze sowie Startwerte

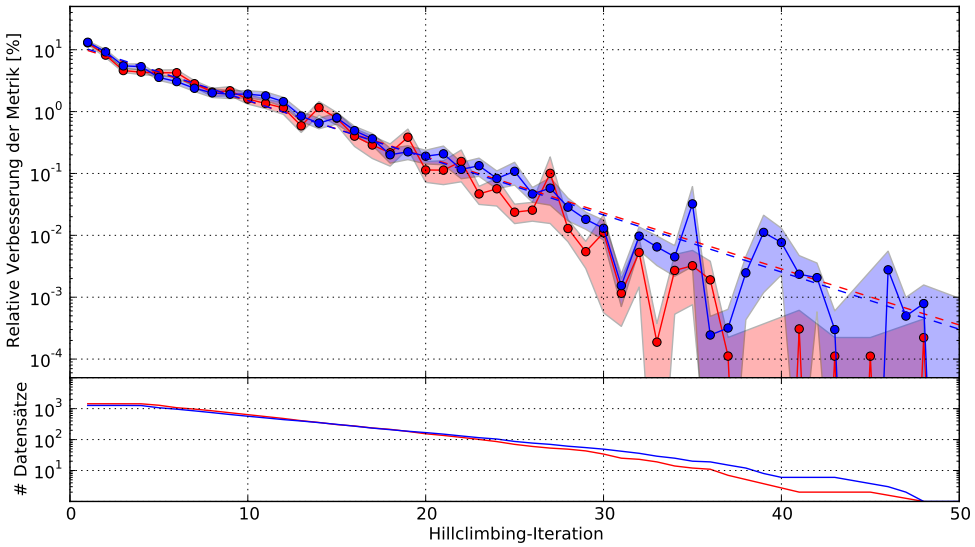


Abbildung IV-4.: Relative Verbesserung pro Hillclimbing-Iteration, gemittelt über alle Datensätze sowie Startwerte für beide Varianten des Hillclimbing-Algorithmus. Sowohl für den iterativen Richtungsmodus (—●—) als auch für den parallelen Richtungsmodus (—●—) wurde ein Fit mit der in Gleichung IV-8 beschriebenen Funktion durchgeführt. Die resultierenden Werte sind Tabelle IV-1 zu entnehmen. Bei der Interpretation der Schwankungen für eine große Anzahl an Iterationen ist zu beachten, dass immer weniger Datensätze vorhanden sind, da das Verfahren bereits zuvor aufgrund der Abbruchbedingung (3 Iterationen ohne Verbesserung) terminiert.

Die experimentell ermittelten Werte für a, b sowie die daraus berechnete maximale Gesamtverbesserung sind in der nachfolgenden Tabelle IV-1 für die verschiedenen Variationen des Hillclimbing-Verfahrens aufgelistet.

Intuitiv würde man erwarten, dass die iterative Variante zu einer schnelleren Konvergenz führt, als die parallele, da bei dieser pro Hillclimbing-Iteration alle Parameter variiert werden, und nicht nur einer. Wie in Tabelle IV-1 zu sehen, ist jedoch kein signifikanter Unterschied der Parameter a oder b festzustellen. Auch die relativen Verbesserungen stimmen im Rahmen der Fehler überein. Demnach ist es also näherungsweise gleich effizient, ob man pro Hillclimbing-Iteration nur einen Neuronparameter oder jeden Parameter verändert. Dies lässt darauf schließen, dass in den meisten Fällen nur einer der Parameter eine große Änderung des Viktor-Purpura-Maßes verursacht, während die anderen Parameter das Maß nicht signifikant verbessern können. Dieses Wissen ist vor allem für eine Implementierung des Hillclimbing-Verfahrens in Software nützlich, da sich die parallele Variante besser paralellisieren lässt als die iterative (siehe Abschnitt IV.1).

Tabelle IV-1.: Durchschnittliche Anzahl an Hillclimbing-Iterationen pro Startwert, Fitparameter der Gleichung IV-8 für die Daten aus Abbildung IV-4, sowie relative Verbesserung für verschiedene Varianten des Hillclimbing-Algorithmus (siehe Abbildung IV-1).

		paralleler Richtungsmodus	**iterativer Richtungsmodus**
Mittlere Anzahl an Iterationen		11.7 ± 6.8	12.1 ± 7.6
Fitparameter der Gleichung IV-8	a [%]	$(12.0 \pm 0.7)\%$	$(12.7 \pm 0.6)\%$
	b	0.812 ± 0.007	0.808 ± 0.006
rel. Verbesserung	experimentell	$(140 \pm 5)\%$	$(144 \pm 6)\%$
	maximal	$(165 \pm 6)\%$	$(168 \pm 5)\%$

IV.4.2. Ermittlung der Konvergenzgeschwindigkeit

Anschließend wurde eine Untersuchung hinsichtlich der Konvergenzgeschwindigkeit durchgeführt. Eine möglichst hohe Konvergenzgeschwindigkeit ist vor allem dann wichtig, wenn keine Parallelisierung der Iterationen möglich ist, und sämtliche Simulationen nacheinander durchgeführt werden müssen.

Zur Ermittlung der Konvergenzgeschwindigkeit betrachten wir jeweils die Differenz des Viktor-Purpura-Maßes der insgesamt besten gefundenen Lösung S_{max} und der besten Lösung nach k Simulationen $S_{\text{sim}}(k)$. Diese Werte sind für die PSO sowie für das Hillclimbing-Verfahren in Abbildung IV-5 aufgetragen.

Zur weiteren Untersuchung wurde auch hier für beide Datensätze ein Fit durchgeführt. Für das Hillclimbing-Verfahren eignet sich offensichtlich ein Fit mit einer Funktion der Form

$$S_{\text{max}} - S_{\text{sim}}(k) = c \cdot d^k, \tag{IV-10}$$

wobei c die anfängliche Differenz des Viktor-Purpura-Maßes von der besten gefundenen Lösung darstellt, und $d < 1$ der relativen Annäherung an diese pro durchgeführter Simulation entspricht. Dieser Zusammenhang erscheint auch aus theoretischer Sicht plausibel, denn je weiter man sich dem globalen Optimum angenähert hat, desto geringer fällt der Unterschied bei nachfolgenden Iterationen aus. Beim PSO-Verfahren dagegen ist die Dynamik der Partikel stark zufallsbedingt - der Verlauf in Abbildung IV-5 lässt sich deshalb schlechter mit Gleichung IV-10 fitten. Die erhaltenen Fitparameter sind in Tabelle IV-2 zu sehen.

Hierbei fällt vor allem auf, dass sich das Hillclimbing- und PSO-Verfahren bezüglich der Konvergenzgeschwindigkeit relativ ähnlich verhalten. Nach den Fitparametern konvergiert für die hier verwendeten Testszenarios das Hillclimbing-Verfahren etwas schneller, wie man jedoch anhand der Abbildung sieht, ist die Geschwindigkeit beim PSO-Verfahren weniger kontinuierlich. Um eine genauere Aussage zu treffen müssten noch weitere Simulationen durchgeführt werden.

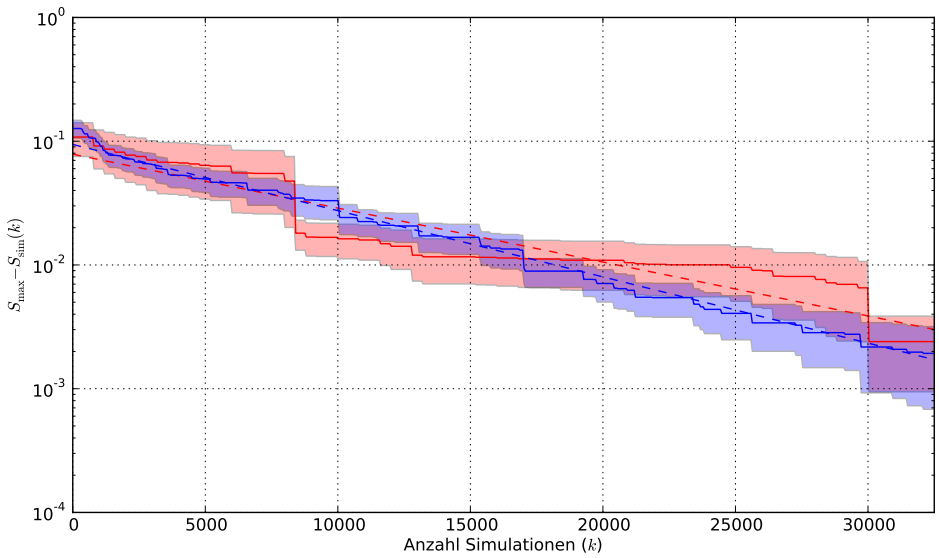


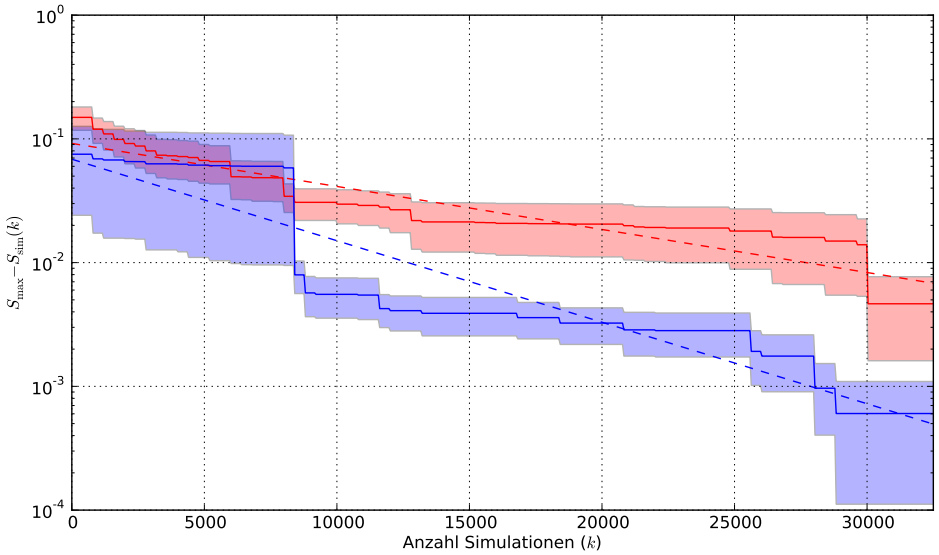
Abbildung IV-5.: Mittlere Differenz zwischen dem besten gefunden Wert der VP-Metrik und den besten Werten nach k Simulationen für das Hillclimbing-Verfahren (——) sowie die PSO (——). Beim PSO-Verfahren wurde der beste Wert jeweils nach einer PSO-Iteration (d.h. 400 Simulationen) ausgelesen, beim Hillclimbing-Verfahren der bisherig beste Wert nach Erreichen des Abbruchkriteriums für einen Startwert (Anzahl der Simulationen skaliert mit der Anzahl der benötigten Iterationen). Zur nachfolgenden Auswertung wurde zusätzlich ein Fit mit Gleichung IV-10 durchgeführt, die resultierende Werte sind Tabelle IV-2 zu entnehmen.

IV.4.3. Einfluss der Startparameter auf die Konvergenzgeschwindigkeit

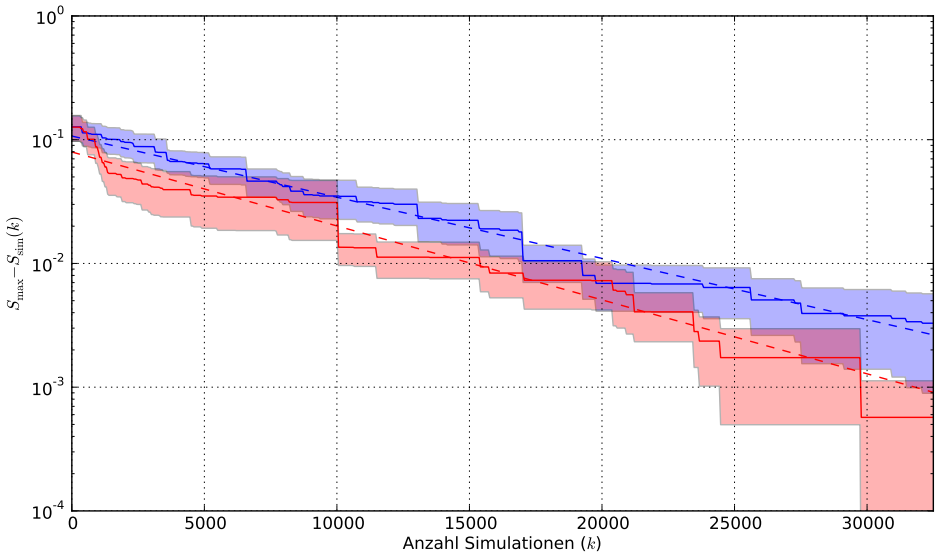
Bisher wurden die beiden Algorithmen im Allgemeinen miteinander verglichen - es erscheint jedoch plausibel, dass eine geeignete Wahl der Startparameter dazu führt, dass die Algorithmen schneller konvergieren, und somit nach einer geringeren Zeit die beste Lösung vorliegt.

Zur Überprüfung dieser Theorie wurde analog zu oben die Differenz des Viktor-Purpura-Maßes der besten gefundenen Lösung zu der Lösung nach k Simulationen aufgetragen. Dies ist für die PSO sowie für das Hillclimbing-Verfahren in Abbildung IV-6 zu sehen. Analog zu oben wurde wieder ein Fit durchgeführt, die ermittelten Fitparameter sind in Tabelle IV-2 zu sehen.

Während das Verfahren zur Abschätzung der Neuronparameter (siehe Abschnitt IV.3) im Falle der PSO für die ausgewerteten Testszenarios zu einer deutlichen Beschleunigung führte, hatte es auf die Konvergenz beim Hillclimbing-Verfahren keinen oder sogar einen negativen Einfluss. Dies liegt wahrscheinlich daran, dass sich die tatsächlichen Parameter in einigen Fällen außerhalb der vom Algorithmus zurückgelieferten Normalverteilung befinden, z.B. weil die



(a) Partikel-Schwarm-Optimierung



(b) Hillclimbing-Verfahren

Abbildung IV-6.: Differenz zwischen dem besten gefunden Wert der VP-Metrik und den besten Werten nach k Simulationen für die Partikelschwarm-Optimierung sowie das Hillclimbing-Verfahren. Die Schaubilder vergleichen jeweils die Gitter-Methode zur Auswahl der Startparameter (——) mit dem vorgestellten Verfahren zur Abschätzung der Neuronparameter (——). Zum Vergleich wurde jeweils ein Fit mit der in Gleichung IV-10 genannten Funktion durchgeführt, die Ergebnisse sind in Tabelle IV-2 zu sehen.

Tabelle IV-2.: Fitparameter zu den Abbildungen IV-5 und IV-6

		c	d
Abbildung IV-5	PSO	(0.079 ± 0.002)	$1 - (100 \pm 1) \cdot 10^{-6}$
	Hillclimbing	(0.0942 ± 0.0008)	$1-(123.2\pm0.4)\cdot10^{-6}$
PSO, Abbildung IV-6	Gitter	(0.092 ± 0.002)	$1 - (80 \pm 1) \cdot 10^{-6}$
	Abschätzung der NP	(0.069 ± 0.004)	$1 - (152 \pm 3) \cdot 10^{-6}$
HC, Abbildung IV-6	Gitter	(0.080 ± 0.001)	$1 - (138 \pm 1) \cdot 10^{-6}$
	Abschätzung der NP	(0.107 ± 0.001)	$1 - (114 \pm 7) \cdot 10^{-6}$

verwendeten Näherungen (siehe Abschnitt VII.2) für den Datensatz zu ungenau sind. Während das Hillclimbing-Verfahren lokal begrenzt sucht und somit nur lokale Maximas finden kann, ist das PSO-Verfahren robuster und erkundet durch die Trägheit der Partikel und durch Zufallskomponenten auch weiter vom Startpunkt entfernte Neuronparameter. Dieser Effekt lässt sich ggf. vermeiden, wenn man einen größeren Fehler der vom Abschätzungs-Algorithmus zurückgelieferten Werte annimmt, oder aber eine exakte Simulation statt der verwendeten Näherungsverfahren nutzt.

IV.4.4. Genauigkeit der Parameter bei der PSO

Bisher wurden lediglich gewisse Eigenschaften untersucht, welche die Geschwindigkeit der Verfahren betreffen - ein weiterer entscheidender Punkt ist jedoch, wie gut die gefundenen Parameter in Abhängigkeit vom verwendeten Algorithmus mit den gesuchten Referenzparametern übereinstimmen.

Für verschiedene Methoden zur Wahl der Startparameter wurde jeweils die mittlere relative Abweichung der ermittelten Fitparameter von den tatsächlichen Neuronparametern berechnet. Die Ergebnisse sind in Tabelle IV-3 zusammengefasst.

Beim Vergleich der Ergebnisse für die verschiedenen Methoden zur Wahl der Startparameter zeigte sich, dass die Verwendung eines Gitters zur Ermittlung der Startparameter offensichtlich einige Vorteile gegenüber dem Ziehen aus Normalverteilungen bietet. Dadurch, dass bei dem Gitter sämtliche Kombinationen abgedeckt sind, ist die Wahrscheinlichkeit größer, dass auch ein Parametersatz existiert, der bereits in der Nähe der gesuchten Lösung liegt, was bei einer Normalverteilung nicht sichergestellt ist.

Verwendet man dagegen den oben vorgestellten Algorithmus zur Abschätzung der Neuronparameter, so lässt sich der relative Fehler im Durchschnitt noch weiter reduzieren. Da jedoch keine exakte Simulation zum Einsatz kommt, sondern nur eine Näherungslösung (siehe Abschnitt VII.2), sind die Startparameter nicht in jedem Fall gleich gut gewählt. Während beim Stimulus *Einfache Stufen* die Annäherung mittels zeitlich konstanter Ströme sehr gut funktioniert, ist dies beim *postsynaptischen Strom* nicht der Fall - dennoch konnte auch hier im

Tabelle IV-3.: Vergleich der Ergebnisse des PSO-Algorithmus für unterschiedliche Methoden zur Wahl der Startparameter

		Normalverteilung	**Gitter**	**Abschätzung der NP**
unvollständige Neuronparam.		0%	0%	0%
Durchschnitt	rel. Fehler	$155 \pm 19\%$	$129 \pm 36\%$	$80 \pm 63\%$
	VP Maß[1]	0.08 ± 0.03	0.25 ± 0.09	0.19 ± 0.27
	RMS Maß[1]	0.11 ± 0.04	0.15 ± 0.07	0.26 ± 0.25
Einfache Stufen	rel. Fehler	135%	110%	0.0003%
	VP Maß	0.12	0.00	1.00
	RMS Maß	0.10	0.14	1.00
Postsyn. Strom	rel. Fehler	177%	88%	31%
	VP Maß	0.10	0.27	0.15
	RMS Maß	0.15	0.21	0.13

[1] Bei dem durchschnittlichen VP/RMS-Maß sind nur erfolgreiche NEST-Simulationen beinhaltet.

Tabelle IV-4.: Vergleich der Ergebnisse des Hillclimbing-Algorithmus für unterschiedliche Methoden zur Wahl der Startparameter

		Normalverteilung	**Gitter**	**Abschätzung der NP**
unvollständige Neuronparam.		0%	0%	0%
Durchschnitt	rel. Fehler	$132 \pm 43\%$	$128 \pm 40\%$	$104 \pm 52\%$
	VP Maß[1]	0.08 ± 0.04	0.27 ± 0.10	0.27 ± 0.15
	RMS Maß[1]	0.10 ± 0.07	0.16 ± 0.08	0.21 ± 0.11
Einfache Stufen	rel. Fehler	107%	94%	96%
	VP Maß	0.12	0.00	0.50
	RMS Maß	0.14	0.18	0.05
Postsyn. Strom	rel. Fehler	72%	78%	58%
	VP Maß	0.16	0.31	0.26
	RMS Maß	0.25	0.23	0.23

[1] Bei dem durchschnittlichen VP/RMS-Maß sind nur erfolgreiche NEST-Simulationen beinhaltet.

Vergleich zur Verwendung eines Gitters der relative Fehler der Parameter deutlich reduziert werden, und das, obwohl die Viktor-Purpura-Metrik eigentlich auf eine schlechtere Lösung hindeutet (siehe Tabelle IV-3).

IV.4.5. Genauigkeit der Parameter beim Hillclimbing-Verfahren

Analog zum vorherigen Abschnitt wurde auch beim Hillclimbing-Verfahren überprüft, mit welcher Methode zur Wahl der Startparameter die besten Neuronparameter gefunden wurden. Die Ergebnisse sind in Tabelle IV-4 zusammengefasst.

Hierbei zeigt sich, dass der relativen Fehler nahezu unabhängig von der Methode der Startparameter ist. Sowohl die Verwendung eines Gitters zur Erzeugung der Startparameter als auch der Algorithmus zur Abschätzung der Neuronparameter verringerte die Fehler nur gering. Grund hierfür ist wahrscheinlich, dass die Reichweite im Parameterraum so gering ist, dass z.B. auch bei Verwendung eines Gitters nicht der gesamte Parameterbereich abgedeckt wird. Dadurch, dass sich die einzelnen Durchläufe für verschiedene Startwerte nicht gegenseitig beeinflussen, ist es hauptsächlich Zufall, wenn sich ein Startwert in unmittelbarer Nähe des globalen Maximums befindet, sodass eine Konvergenz zu diesem möglich ist.

Abbildung IV-7 visualisiert die Ergebnisse verschiedener Methoden zur Wahl der Startwerte beider Algorithmen in Abhängigkeit des verwendeten Stromstimulus.

IV.4.6. Abhängigkeit der Ergebnisse vom verwendeten Stromstimulus

Bei beiden Algorithmen sind die Resultate wie erwartet nahezu unabhängig vom verwendeten Stromstimulus. Dies wird auch in der nachfolgenden Abbildung IV-8 noch einmal deutlich, welche die Ergebnisse sämtlicher durchgeführter Simulationen miteinander vergleicht.

Eine deutlich größere Abhängigkeit vom jeweiligen Strom konnte dagegen beim Algorithmus zur Approximation der Neuronparameter festgestellt werden. Die Ursachen hierfür liegen wahrscheinlich an den verwendeten Näherungen, welche konstante Ströme besser beschreiben können, als z.B. die generierten postsynaptischen Ströme. Trotz der verwendeten Näherungen konnte der relative Fehler bei allen Datensätzen gesenkt werden. Um diese Erklärung zu verifizieren wären weitere Überprüfungen notwendig, bei denen das Näherungsverfahren durch eine exakte Simulation ersetzt wird.

IV.5. Zusammenfassung

In diesem Kapitel wurden zwei Algorithmen analysiert, welche eine Anpassung der Neuronparameter ausschließlich mithilfe der von einer Metrik zurückgelieferten Bewertung vornehmen. Durch Verwendung der *Viktor-Purpura*-Metrik sind dazu lediglich die Spikezeitpunkte sowie eine Stimulation des Neurons (Strom bzw. Spikes) notwendig.

Im ersten Teil von Abschnitt IV.4 wurde zunächst verifiziert, dass der Hillclimbing-Algorithmus wie gewünscht die anfänglichen Werte der Metrik systematisch verbessert. Hierbei konnte kein signifikanter Unterschied zwischen der iterativen und der parallelen Abarbeitung verschiedener Richtungskomponenten festgestellt werden. Bei der parallelen Variante konnten die anfänglichen Maße im Schnitt um $(140 \pm 5)\%$ verbessert werden. Daraus wird erkennbar,

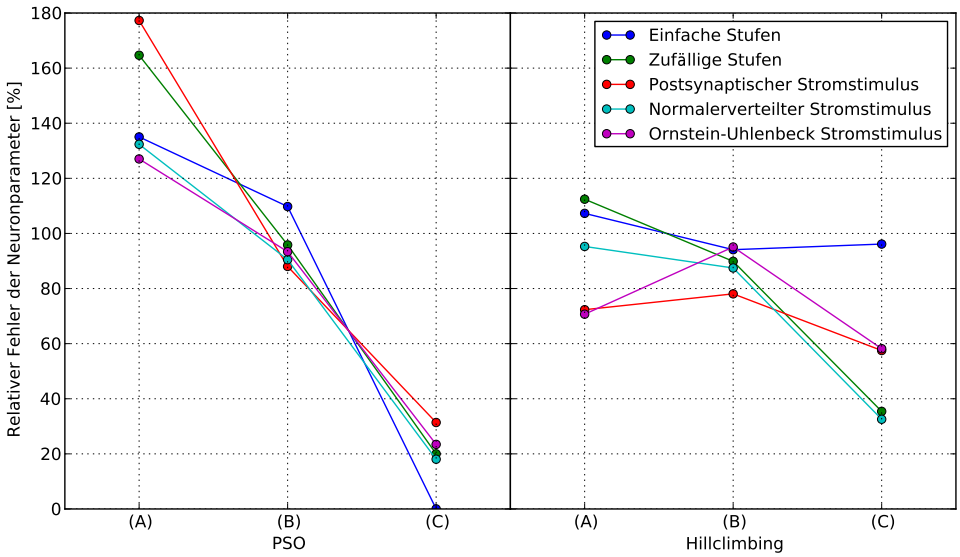


Abbildung IV-7.: Relativer Fehler der Neuronparameter für die PSO sowie das Hillclimbing-Verfahren in Abhängigkeit der verwendeten Methode zur Wahl der Startparameter. Als Referenz wurden die Parameter `adaption (1)` aus Tabelle II-2 verwendet. Getestet wurden die Methoden *Startparameter aus einer vordefinierten Normalverteilung* (A), *Startparameter aus einem Gitter* (B) und das *Verfahren zur Abschätzung der Neuronparameter* (C).

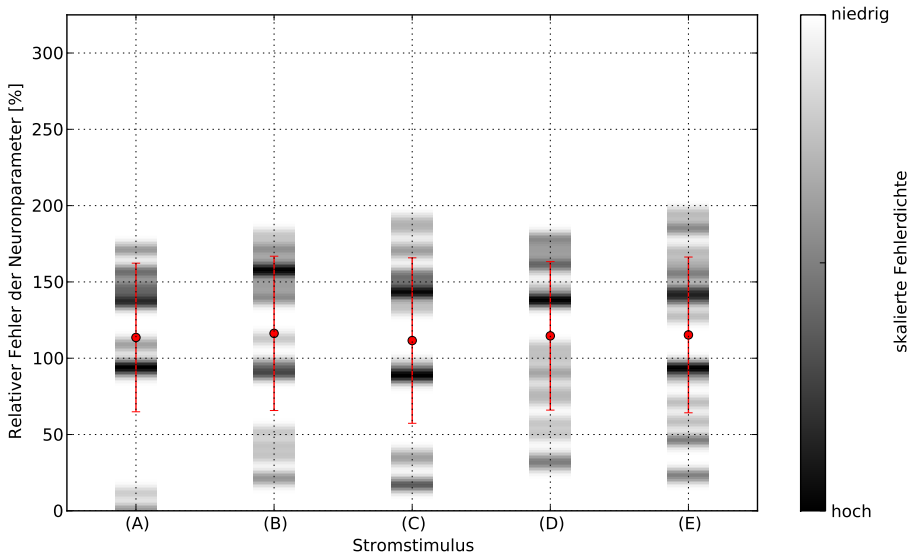


Abbildung IV-8.: Relativer Fehler der Neuronparameter in Abhängigkeit des verwendeten Stromstimulus. Der Farbverlauf veranschaulicht die Häufigkeit der relativen Fehler über alle ausgewerteten Datensätze, zusätzlich wurde jeweils der mittlere Fehler (•) sowie die Standardabweichung eingezeichnet. Die verwendeten Stromstimuli sind der Strom *Einfache Stufen* (A), *Zufällige Stufen* (B), *Normalverteilter Stromstimulus* (C), *Postsynaptischer Stromstimulus* (D) und *Ornstein-Uhlenbeck Stromstimulus* (E). Der mittlere relative Fehler ist bei jedem Stromstimulus ungefähr gleich groß.

dass der Hillclimbing-Algorithmus wie gewünscht funktioniert. Andererseits wird auch deutlich, dass bereits eine relativ gute Lösung vorliegen muss, um hohe Absolutwerte zu erhalten. Unter gewissen Annahmen wurde außerdem die maximal mögliche Verbesserung abgeschätzt, wenn man beliebig viele Hillclimbing-Iterationen durchführen würde, um das Ergebnis immer weiter zu optimieren. Damit wäre theoretisch eine Verbesserung von bis zu etwa $(165 \pm 6)\%$ erreichbar.

Bei der Überprüfung der Konvergenzgeschwindigkeit konnte kein signifikanter Unterschied zwischen der PSO und dem Hillclimbing-Verfahren festgestellt werden. Aufgrund der Zufallskomponente bei der PSO wäre es jedoch notwendig, noch weitere Testläufe durchzuführen, um einen verlässlichen Mittelwert zu erhalten. Während das Verfahren zur Abschätzung der Neuronparameter (siehe Abschnitt IV.3) beim Hillclimbing-Algorithmus keinen großen Einfluss auf die Konvergenzgeschwindigkeit zeigte, führte es beim PSO-Verfahren im Schnitt zu einer schnelleren Konvergenz.

Abschließend wurde noch überprüft, welche Methode zur Wahl der Startparameter bei beiden Algorithmen zu kleineren Fehlern der Neuronparameter führte. Bei beiden Algorithmen zeigte sich, dass eine Wahl der Parameter mittels Normalverteilung zu schlechteren Ergebnissen führt. Dies liegt vermutlich daran, dass bei einer Normalverteilung nicht sichergestellt ist, dass der gesamte Parameterraum abgedeckt wird - man könnte dem zwar entgegenwirken, indem man die Anzahl der Partikel / Durchläufe des HC-Verfahrens erhöht, was aber in jedem Falle mit einer höhen Laufzeit verbunden wäre. Beim Hillclimbing-Algorithmus zeigte das Verfahren zur Abschätzung der Neuronparameter nahezu keinen Effekt, beim PSO-Verfahren dagegen konnten damit sämtliche Ergebnisse noch einmal verbessert werden.

Da beide Verfahren jeweils nach einer festgelegten Anzahl an Iterationen (bei der PSO) bzw. Startwerten (bei Hillclimbing) abbrechen, und die Anzahl der Startwerte bzw. Anzahl der Partikel näherungsweise gleich gewählt wurde, stimmen die Laufzeiten beider Algorithmen ungefähr überein und betragen auf der verwendeten Hardware[6] im Mittel ca. 1 Std (bei Verwendung von durchschnittlich 2 Cores). Der Algorithmus zur Abschätzung der Neuronparameter erhöht diese Laufzeit allerdings noch einmal um eine ca. 1 Std (bisher nur Verwendung eines Cores), was darauf zurückzuführen ist, dass die in Abschnitt VII.2 vorgestellte Näherungslösung direkt in Python implementiert wurde.

Bezüglich der in dieser Untersuchung getesteten Referenzdaten lieferte das PSO-Verfahren in Kombination mit dem Algorithmus zur Abschätzung der Neuronparameter insgesamt die besten Ergebnisse.

IV.5.1. Weiterführende Untersuchungen

Auch nach den obigen Überprüfungen kann die Untersuchung der Verfahren noch nicht als abgeschlossen betrachtet werden. Basierend auf den ermittelten Problemen stellt sich die Frage, wie sich die Algorithmen und deren Varianten modifizieren lassen, um die Konvergenzgeschwindigkeit bzw. die erreichten Metriken weiter zu erhöhen. Prinzipiell kann man natürlich bei beiden Algorithmen ein besseres Ergebnis erzielen, indem man die Anzahl der Partikel / Durchläufe erhöht, um früher oder später einen Glückstreffer zu landen. Diese Methode wäre jedoch nicht praktikabel für Anwendungen in der Praxis.

[6] AMD Phenom(tm) II X4 965 Processor, 4x 3400 Mhz

Auch eine Modifikation der Abbruchbedingungen wäre möglich, es ist jedoch fraglich, ob damit signifikant bessere Ergebnisse zu erzielen sind.

Insgesamt zeigte sich während der Analyse der Daten, dass unter Verwendung der Viktor-Purpura-Metrik in den meisten Fällen keine Konvergenz zum globalen Maximum stattfindet. Ein möglicher Ansatzpunkt für Verbesserungen wäre die Überprüfung, ob dies ggf. bei anderen Metriken der Fall ist. Eine Bewertung weiterer Metriken bezüglich ihrer Eignung in Optimierungsverfahren wurde z.B. in Born (2012) durchgeführt. Meiner Ansicht nach wird jedoch jede ausschließlich auf Spike-Zeitpunkten basierende Metrik das Problem haben, dass die Zeiten nicht eindeutig zugeordnet werden können, und somit keine Konvergenz zum globalen Maximum möglich ist. Der hier vorgestellte Algorithmus zur Abschätzung der Neuronparameter stellt einen neuen Ansatz dar, in welchem durch schrittweise Simulation mehr Informationen gewonnen und die Ergebnisse (zumindest beim PSO-Verfahren) deutlich verbessert werden konnten.

Abgesehen von der Verbesserungen der Fitalgorithmen ist allerdings noch zu untersuchen, inwiefern die verwendeten Neuronparameter überhaupt wieder vollständig rekonstruiert werden können, oder ob für bestimmte Referenzspikes mehrere vollständig äquivalente Parametersätze existieren. Bei einer vollständigen Beschreibung eines Neurons mittels Membranpotential wurde in Paninski et al. (2004) gezeigt, dass in diesem Fall die besten Neuronparameter eindeutig festgelegt sind, dies muss bei der ausschließlichen Angabe der Spikezeiten aber nicht zwangsläufig erfüllt sein. Falls verschiedene Parameter grundsätzlich zu identischen Spikes führen, könnten entsprechende Erkenntnisse auch dazu verwendet werden, die Anzahl der Parameter des Modells weiter zu reduzieren. Dabei wird zwar der direkte Zusammenhang zum biologischen Membranpotential aufgegeben, andererseits wäre aber eventuell eine höhere Konvergenzgeschwindigkeit bzw. geringere Anzahl an Simulationen möglich.

V. Anwendungen der Optimierungsverfahren

Bisher wurde jeweils ein direkt in das Neuron injizierter Strom zur Stimulation verwendet, der unabhängig vom aktuellen Membranpotential ist. In der Natur erfolgen die Anregungen externer Neuronen jedoch vorwiegend über Synapsen, welche zu einem membranpotentialabhängigen Strom und somit zu einer anderen Dynamik führen (siehe Gleichung II-7). Wir werden uns im Folgenden auf den Fall der Stimulation durch eine einzelne Synapse beschränken und qualitativ untersuchen, ob die Algorithmen auch dann anwendbar sind, und zu vergleichbaren Ergebnissen führen wie beim Stromstimulus.

Zudem wurden die Algorithmen exemplarisch an Datensätzen für drei mögliche Anwendungen getestet. Bei allen bisherigen Überprüfungen kamen jeweils künstlich erzeugte Referenzdaten zum Einsatz, um die Algorithmen vergleichen zu können. Zum Test der Praxistauglichkeit der Algorithmen wurden Fits an biologischen Referenzdaten, an Daten anderer Neuronmodellen sowie an einer Hardwaremessung durchgeführt.

V.1. Spikestimulus statt Stromstimulus

Der in Kapitel III beschriebene Algorithmus basierend auf einer linearen Regression kann in der aktuellen Implementierung noch nicht dazu genutzt werden, um auch eine Anpassung mittels Spikestimulus vorzunehmen. Diese Möglichkeit könnte in zukünftigen Versionen ergänzt werden, indem man Gleichung II-7, welche den Zusammenhang zwischen eintreffenden Spikes und Strom beschreibt, in Gleichung III-1 einsetzt. Damit das Gleichungssystem linear bleibt, müsste der Parameter τ_k dann analog zu τ_w durch Minimierung der Residuen ermittelt werden. Im Rahmen dieser Arbeit wurde diese Idee jedoch nicht weiter verfolgt, weshalb wir uns hier auf die PSO sowie den Hillclimbing-Algorithmus beschränken.

Um das oben beschriebene Szenario der Anregung eines Neurons durch eine einzelne Synapse zu testen, wurde zunächst mithilfe einer Simulation eine Folge von Spikes generiert. Für die Simulation kamen die Neuronparameter des Datensatzes `adaption (1)` aus Tabelle II-2 sowie der *Postsynaptische Stromstimulus* zum Einsatz. Für die nachfolgenden Fits werden nur noch die so erzeugten Spikes benötigt, deren Feuerrate 14.7 Hz betrug.

Die Referenzdatensätze wurden mit den Neuronparametern `tonic spiking (0)` und `adaption (1)` erzeugt, welche um die Parameter der synaptischen Verbindung ergänzt wurden.

$$E_\mathrm{k} = 0.0\ \mathrm{mV}, \tau_\mathrm{k} = 5.0\ \mathrm{ms}, \mu_\mathrm{k} = 0.05\ \mathrm{nS} \tag{V-1}$$

Mithilfe der synaptischen Parameter wurden ähnlich zur bisherigen Vorgehensweise Referenzdatensätze durch Simulation erzeugt, und anschließend versucht mit Hilfe der Fitalgorithmen die ursprünglichen Parameter wieder zu rekonstruieren.

V.1.1. PSO-Algorithmus

Für beide Referenzdatensätze wurden jeweils zwei Fits mit der PSO durchgeführt. Zum Einen wurden die Startparameter aus einem Gitter gewählt, zum Anderen wurde der Algorithmus zur Abschätzung der Neuronparameter (siehe Abschnitt IV.3) getestet. Die Ergebnisse der Fits sind in der nachfolgenden Tabelle V-1 zu sehen.

Tabelle V-1.: Ergebnisse der PSO für zwei verschiedene Referenzparameter unter Verwendung von Spikestimulus statt Stromstimulus

		Gitter	**Abschätzung der NP**
`tonic spiking (0)`	rel. Fehler	156%	137%
	VP Maß	0.81	0.25
	RMS Maß	0.23	0.05
`adaption (1)`	rel. Fehler	136%	0.002%
	VP Maß	0.32	1.00
	RMS Maß	0.07	1.00

Durch Verwendung des Verfahrens zur Abschätzung der Neuronparameter konnte bei beiden Datensätzen der relative Fehler reduziert werden. Während bei dem Strom `adaption (1)` eine sehr gute Übereinstimmung der Parameter erzielt werden konnte (VP $= 1.00$), ist bei `tonic spiking (0)` die Verbesserung nur gering. Ein möglicher Grund ist, dass in letzterem Fall das verwendete Näherungsverfahren (siehe Abschnitt VII.2) deutlich von der exakten Lösung abweicht, weil die Spikerate besonders hoch ist (der zeitliche Abstand zweier Spikes ist in gewissen Abschnitten lediglich ≈ 1.5ms). Dadurch dass das Näherungsverfahren nicht mehr anwendbar ist scheitert die Abschätzung, weshalb in diesem Fall das Gitter zur Erzeugung der Startparameter die besseren Viktor-Purpura-Maße liefert.

V.1.2. Hillclimbing-Algorithmus

Analog zu Unterabschnitt V.1.1 wurden für die gleichen Referenzdaten Fits mit dem Hillclimbing-Algorithmus durchgeführt, deren Resultate in Tabelle V-2 zusammengefasst sind.

Während auch hier bei `adaption (1)` der relative Fehler durch den Algorithmus zur Abschätzung der Neuronparameter stark minimiert werden konnte, war im Falle von `tonic spiking (0)` keine Verbessung möglich.

V.1.3. Fazit

Bezüglich der getesteten Szenarien, scheinen die Verfahren vergleichbar gute Ergebnisse zu liefern, wie bei Verwendung eines Stromstimulus. Dennoch zeigt sich auch hier, dass trotz

Tabelle V-2.: Ergebnisse des Hillclimbing-Algorithmus für zwei verschiedene Referenzparameter unter Verwendung von Spikestimulus statt Stromstimulus

		Gitter	**Abschätzung der NP**
`tonic spiking (0)`	rel. Fehler	146%	182%
	VP Maß	0.82	0.74
	RMS Maß	0.15	0.09
`adaption (1)`	rel. Fehler	169%	12%
	VP Maß	0.46	0.74
	RMS Maß	0.02	0.39

größerem relativen Fehler in einigen Fällen ein besseres Viktor-Purpura-Maß erreicht werden konnte. Die schlechten Ergebnisse bei Verwendung der Referenzparameter `tonic spiking (0)` und vorheriger Abschätzung der Neuronparameter liegt wahrscheinlich an der besonders hohen Feurrate des Neurons sowie dem stark fluktuierenden Strom, wodurch die Näherungsverfahren nicht mehr anwendbar sind. Dieses Problem würde bei einer exakten Simulation oder der Verwendung einer Hardware-Simulation nicht auftreten. Ein weiteres Problem bei so hohen Feuerraten könnte außerdem auch die feste Wahl des Parameters c_3 (siehe Unterabschnitt IV.3.2) sein, welcher für die Gewichtung der Fehler verantwortlich ist. Durch eine dynamische Wahl dieses Parameters könnte die Abschätzung noch besser an den jeweiligen Referenzdatensatz angepasst werden.

V.2. Daten gemessen an biologischen Neuronen

Um die Anwendbarkeit der linearen Regression bei biologischen Messdaten zu überprüfen, wurden aus den Daten der *Quantitative Single-Neuron Modeling Competition 2009A* (Gerstner & Naud, 2009) einige Strom- sowie Spannungsausschnitte von 5 Sekunden Dauer extrahiert[1]. Diese extrahierten Daten wurden als Eingabe des in Kapitel III beschriebenen Algorithmus verwendet, wobei der Modus *linear interpolierte Bins* (Unterabschnitt III.1.4) zum Einsatz kam. Auf eine Anpassung mit den anderen erläuterten Optimierungsverfahren wurde aus Zeitgründen verzichtet. Ein Beispiel für einen solchen Ausschnitt dieser Daten ist in Abbildung V-1 zu sehen.

Trotz der Überprüfung verschiedener Ausschnitte (sowohl Bereiche mit rauschendem, als auch Bereiche mit konstantem Strom) konnte der Regressionsalgorithmus für keinen der Datensätze die Neuronparameter vollständig extrahieren, was für eine anschließende Simulation

[1]Die Original-Datensätze haben eine Länge von jeweils 40 Sekunden, was ggf. zu einem besseren Fitergebnis führen würde - die Laufzeit wäre jedoch bei der aktuellen Implementierung nicht mehr praktikabel

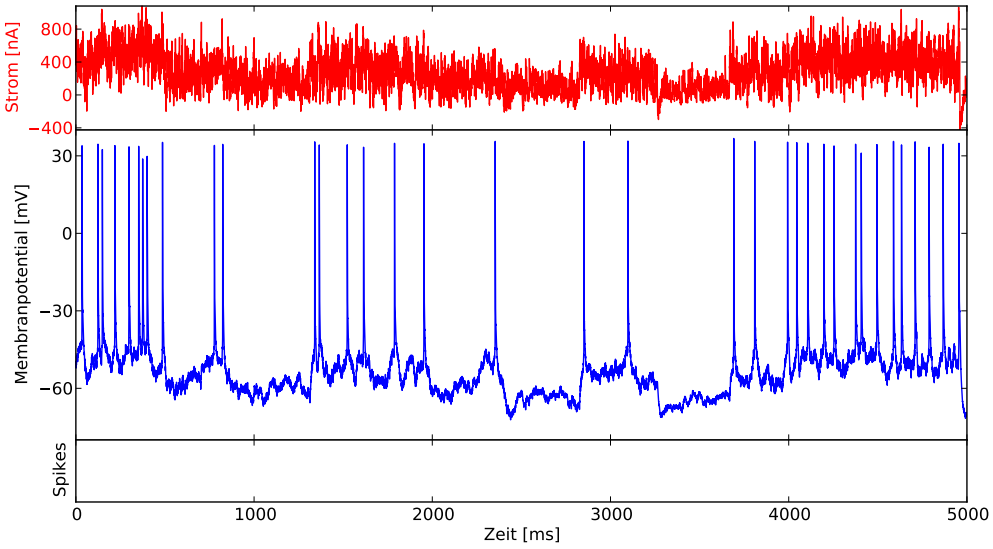
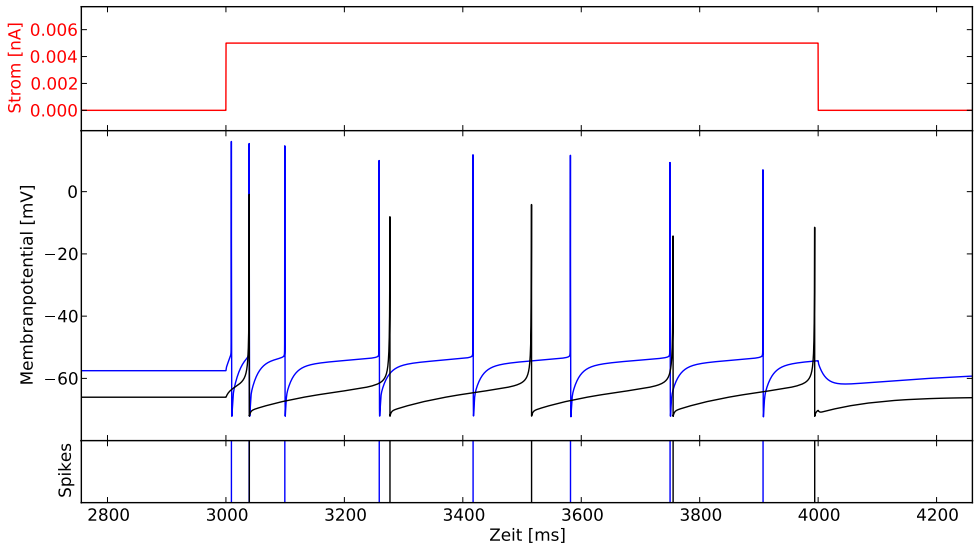


Abbildung V-1.: Einer der für die lineare Regression verwendeten Strom (——) sowie Membranpotential (——)-Ausschnitte. Die Positionen der Spikes wurden von dem Fitalgorithmus aus dem Membranpotential rekonstruiert, und sind deshalb nicht angegeben.

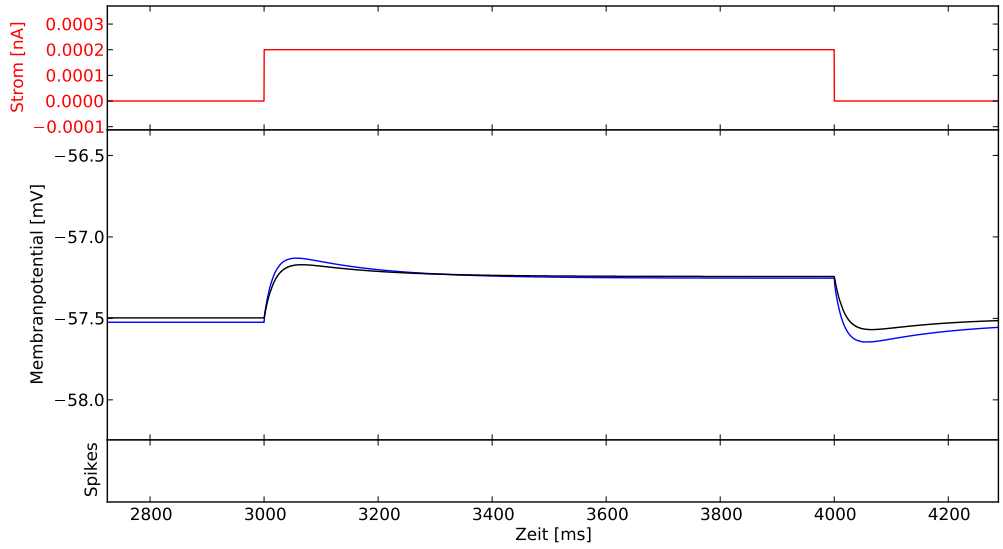
notwendig wäre. Grund hierfür könnte zum Einen sein, dass die Messdaten nicht ausreichend genau dem AdEx-Modell entsprechen, was für die korrekte Funktionalität des Verfahrens Voraussetzung ist, oder aber dass die zusätzlichen Freiheitsgrade der Binparameter (siehe Gleichung III-3) eine Konvergenz zur gewünschten Lösung verhindern. In der jetzigen Version ist das Verfahren offensichtlich noch nicht geeignet, um beliebige Eingabedaten zu verarbeiten. Nicht getestet wurde allerdings der Effekt der anderen vorgeschlagenen Modifikationen (ab Unterabschnitt III.1.5), da diese schon bei künstlich erzeugten Referenzdaten keine signifikante Verbesserung brachten. Außerdem besteht die Möglichkeit durch manuelle Angabe von Zusatzinformationen (z.B. weitere Einschränkung bestimmter Neuronparameter) ein besseres Fitergebnis zu erhalten.

V.3. Daten von anderen Neuronmodellen

Die Anpassung von Daten anderer Neuronmodelle würde es ermöglichen, diese innerhalb gewisser Näherungen ebenfalls auf der BrainScaleS-Hardware zu simulieren. Basierend auf der Arbeit von Lundqvist et al. (2006) wurden Referenzdaten von Neuronen erzeugt, welche die Eigenschaften verschiedener Zellen des Neokortex aufweisen. Dabei kam das in Fransén & Lansner (1998) vorgestellten Modell zum Einsatz, welches ähnlich dem Hodkin-Huxley-Modell verschiedene Ionen separat beschreibt und somit von wesentlich mehr Parametern abhängt, als



(a) RSNP Datensatz 1



(b) RSNP Datensatz 2

Abbildung V-2.: Die Abbildungen zeigen exemplarisch das Membranpotential sowie die Spikes zweier regulär feuernden Nicht-Pyramidalzellen (RSNP-Zellen) (——), welche durch unterschiedliche Stromstimuli (vergleichbar zu Lundqvist et al. (2006)) angeregt wurde (——). Die Ergebnisse der linearen Regression (——) sind ebenfalls eingezeichnet, sie unterscheiden sich jedoch vor allem beim ersten Datensatz deutlich von den Referenzdaten.

es bei dem AdEx-Modell der Fall ist. Mithilfe des im ersten Teil der Arbeit implementierten linearen Regressionsverfahrens wurde experimentell überprüft, ob sich das Membranpotential auch durch das AdEx-Modell beschreiben lässt. Dazu wurden für insgesamt 12 Datensätze (Stromstimulus sowie erwartetes Membranpotential) verschiedener Zelltypen ausgewertet. Da die Referenzdaten jeweils nur wenige oder teilweise gar keine Spikes beeinhalten, sind diese für spikebasierte Optimierungsverfahren nicht geeignet.

Zwei Beispiele für die verwendeten Datensätze, sowie das simulierten Membranpotential unter Verwendung der per Fit ermittelten Neuronparameter sind in Abbildung V-2 zu sehen. Bei der Auswertung aller Fitergebnisse zeigte sich, dass das Verfahren in vielen Fällen keine brauchbaren Ergebnisse lieferte (z.B. feuerte das Neuron bei einigen ermittelten Parametern dauerhaft). Abgesehen von den bisher schon erläuterten Problemen des Linearen-Regression-Verfahrens kommt bei den hier durchgeführten Tests hinzu, dass die Referenzdaten nicht besonders viele Informationen, und vor allem keine Kombination verschiedener Stromstufen beinhalten.

V.4. Daten gemessen an der BrainScaleS-Hardware

Bisher existieren für die BrainScaleS-Hardware nur begrenzte Möglichkeiten zu Kalibration - es ist zwar möglich jeden Parameter einzeln zu kalibrieren, wenn allerdings eine möglichst präzise Übereinstimmung mit dem AdEx-Modell gefordert wird, dann sind diese Methoden ggf. nicht ausreichend. Um die Anwendbarkeit der hier vorgestellten Algorithmen zu testen, wurde in ein einzelnes Neuron ein periodischer Strom injiziert. Das Membranpotential wurde gemessen und basierend auf den Umrechnungsfaktoren der Hardware in die Werte des AdEx-Modells übersetzt.

Insgesamt wurden drei Aufnahmen ausgewertet, zum Einen die Messung des Membranpotentials bei Verwendung einer Stromstufe als Stimulus, eine Messung bei rauschendem Stromstimulus, sowie eine weitere Messung, bei der zeitweise ein Rauschsignal anliegt, und die restliche Zeit gar kein Signal, sodass das Membranpotential wieder abklingen kann. Zum Zeitpunkt der Messung (Juli 2012) war es leider noch nicht möglich, eine zeitliche Synchronisation zwischen Stromstimulus und aufgenommener Membranspannung zu erhalten - aus diesem Grund muss die zeitliche Verschiebung ebenfalls aus den Messdaten ermittelt werden. Dieser Mehraufwand wird bei zukünftigen Messungen wegfallen, sobald eine Ansteuerung der Triggerausgänge in Hard- und Software implementiert wurde.

V.4.1. Ermittlung der zeitlichen Verschiebung

Der Stromstimulus hatte eine Periodenlänge von 412.8 ms, wobei der Strom innerhalb eines Intervalls von jeweils 3.2 ms konstant war. Um Stromstimulus und das gemessene Membranpotential zu synchronisieren, bieten sich verschiedene Möglichkeiten. Zum Einen wäre es möglich, für verschiedene zeitliche Verschiebungen Fits durchzuführen, und anschließend zu prüfen, bei welcher Einstellung die besten Ergebnisse erzielt werden konnten. Da aufgrund der fehlenden Kalibration sowie der Fehleranfälligkeit des Fitalgorithmus jedoch unklar ist, ob überhaupt realistische Werte zurückgeliefert werden, wurde eine alternative Methode zur Ermittlung der zeitlichen Verschiebung verwendet, welche auf Methoden der Statistik basiert.

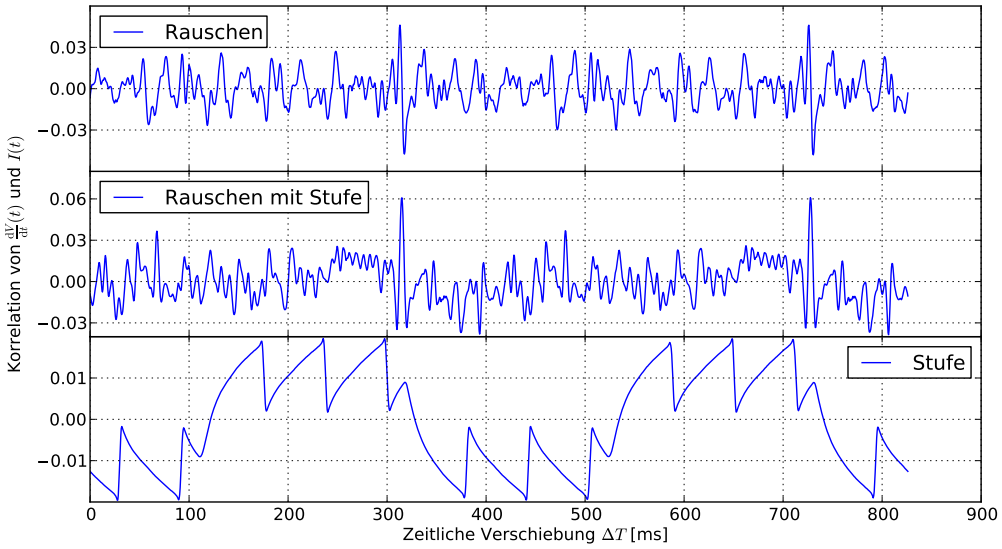


Abbildung V-3.: Korrelation der Daten $I(t)$ und $\frac{dV}{dt}(t+\Delta t)$ dreier Datensätze (Rauschen, Rauschen mit Stufe, Stufe). ΔT. Wie in Unterabschnitt V.4.1 beschrieben, wurde anschließend für die zeitlichen Verschiebungen ΔT, für die die Korrelation ein Maximum annimmt, eine genauere Anpassung durchgeführt (bei den Strömen mit Rauschen nur für Korrelationen größer als 0.027). Anhand der Abbildung wird auch die Periodizität des verwendeten Stromstimulus deutlich.

Anhand der Differentialgleichungen des AdEx-Modells (siehe Gleichung II-1) erwartet man bei korrekter zeitlicher Verschiebung Δt eine Korrelation der Ableitung des Membranpotentials $x_i := \frac{dV}{dt}(t_i+\Delta t)$ mit dem Strom-Input $y_i := I(t_i)$. Die Berechnung der Korrelation erfolgt mithilfe der nachfolgenden Formel:

$$r\left(\frac{dV}{dt}, I\right) = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_j (x_j - \bar{x})^2 \sum_k (y_k - \bar{y})^2} \tag{V-2}$$

Der Koeffizient $r\left(\frac{dV}{dt}, I\right)$ liegt immer zwischen -1 und $+1$, wobei 0 bedeutet, dass die Variablen linear unabhängig sind[2]. Da wir (basierend auf der Formel des AdEx-Modells) nur an positiver Korrelation interessiert sind, sind also die zeitlichen Verschiebungen Δt gesucht, für die $r\left(\frac{dV}{dt}, I\right)$ maximal wird. Durch Anwendung des Verfahrens auf die 3 Datensätze erhält man die in Abbildung V-3 gezeigten Korrelationen.

Anschließend wurde für die zeitlichen Verschiebungen, bei denen die Korrelation ein Maximum erreicht (bei Datensätze mit Rauschen nur für Korrelationen größer als 0.027), jeweils ein Fit mittels linearer Regression sowie dem PSO-Algorithmus ausgeführt.

[2] nichtlineare Abhängigkeiten können mit diesem Verfahren nicht ermittelt werden

V.4.2. Ermittlung der Neuronparameter

Für alle durchgeführten Fits wurde zunächst jeweils das RMS-Maß berechnet, um die tatsächliche zeitliche Verschiebungen zu ermitteln. Dieses Maß eignet sich hier besser als das VP-Maß, da sowohl die Spikes als auch das Membranpotential durch das AdEx-Modell beschrieben werden sollen. Im Folgenden wurde für jeden Datensatz jeweils die Verschiebung angenommen, bei der das RMS-Maß maximiert werden konnte. Bei dem Strom mit einfachen Stufen stimmt dies auch mit der von Hand ermittelten Verzögerung überein, wenn man Anstieg des Stromes sowie den des Membranpotentials aneinander ausrichtet.

Beim Vergleich der drei ermittelten Verzögerungen fällt auf, dass die Werte für alle 3 Datensätze nahezu übereinstimmen, was wahrscheinlich an den Trigger-Bedingungen des zur Messung verwendeten Oszilloskops liegt. In einigen Fällen konnte auch mit einer falschen Verschiebung ein Parametersatz ermittelt werden, der zu einem höheren VP-Maß führt - diese Datensätze wurden für die nachfolgenden Auswertungsschritte jedoch ausgeschlossen, da das Membranpotential deutlich von den Referenzdaten abweicht.

Für die übrigen Datensätze wurde zum Einen das VP-Maß verglichen, zum Anderen wurde die Übereinstimmung zusätzlich manuell überprüft. Die Ergebnisse sind in der nachfolgenden Tabelle V-3 zu sehen.

Tabelle V-3.: Ergebnisse der Fit-Algorithmen bei Anwendung auf drei Hardwaremessungen

	ΔT		**LinReg**	**PSO (Gitter)**	**PSO (Abschätzung der NP)**
Stufen	317.5 ms	VP Maß	0.00	0.24	0.10
		RMS Maß	0.06	0.05	0.03
Rauschen	315.5 ms	VP Maß	0.02	0.16	0.09
		RMS Maß	0.04	0.03	0.02
Rauschen mit Stufe	315.0 ms	VP Maß	0.03	0.23	0.09
		RMS Maß	0.05	0.04	0.06

Bei der manuellen Überprüfung der Ergebnisse zeigte sich, dass die mit dem PSO-Verfahren ermittelten Neuronparameter zwar insgesamt besser die Spikes reproduzieren können, das Membranpotential dafür jedoch deutlich von den Messdaten abweicht. Dies ist auch in Abbildung V-4 zu sehen. Bei Verwendung der linearen Regression stimmt zwar das Membranpotential in gewissen Teilbereichen gut überein, dafür konnte in diesem Fall keine Synchronisierung der Spikezeitpunkte erreicht werden. Ein solches Beispiel ist in Abbildung V-5 zu sehen.

Mithilfe der oben getesteten Algorithmen konnte jedoch kein Satz von Neuronparametern gefunden werden, der eine deutlich bessere Übereinstimmung mit dem Referenzmembranpotential aufweist. Hierbei sind verschiedene Erklärungen denkbar - zum Einen ist es durchaus möglich, dass auch in diesem Fall (wie bei einigen künstlich erzeugten Referenzdaten) die Al-

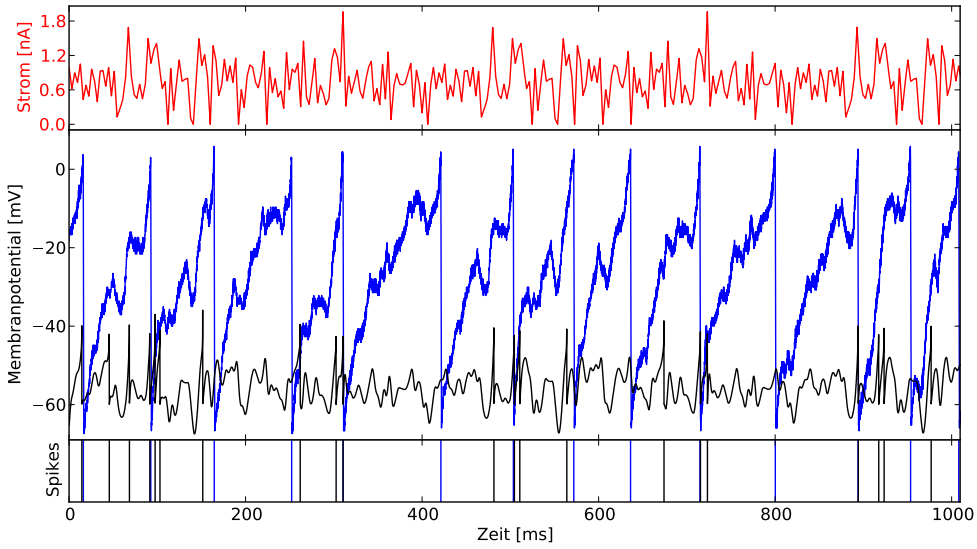


Abbildung V-4.: Stromstimulus mit Rauschen (——) sowie Membranpotential und Spikes der Hardwaremessung (——). Zusätzlich wurde der mit dem PSO-Verfahren ermittelte Fit (VP $= 0.16$) eingezeichnet (——). Sowohl die Spikezeiten als auch das Membranpotential weichen von den Referenzdaten jedoch deutlich ab.

gorithmen scheitern. Da die Hardwaremessungen jedoch vollständig ohne Kalibration durchgeführt wurden, ist es auch möglich, dass sich die gewählten Hardware-Neuronparameter in einem Bereich befanden, in dem sich die Implementierung nicht mehr gemäß des AdEx-Modells verhielt.

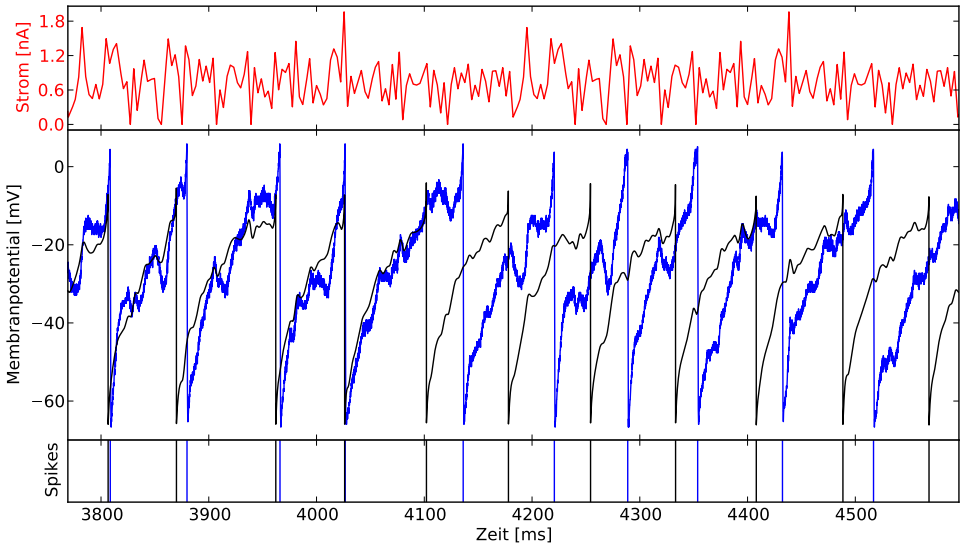


Abbildung V-5.: Stromstimulus mit Rauschen (——) sowie Membranpotential und Spikes der Hardwaremessung (——). In diesem Fall kam das lineare Regressionsverfahren zum Einsatz, das aus den Neuronparametern per Softwaresimulation ermittelte Membranpotential (——) wurde ebenfalls in die Abbildung eingezeichnet.

VI. Diskussion und Ausblick

Ein großes Problem bei der Optimierung von Neuronparametern ist der 11 bzw. 14-dimensionale Parameterraum sowie die Nichtlinearität der Differentialgleichungen des AdEx-Modells. Aufgrund der komplexen Dynamik sind spezielle Verfahren notwendig, um geeignete Neuronparameter aus Referenzdaten (z.B. biologische Messungen oder Hardwaremessungen) zu extrahieren. Im Rahmen dieser Arbeit wurden verschiedene Fitverfahren implementiert und miteinander verglichen. Nach der Evaluation zeigte sich, dass jeder der untersuchten Algorithmen gewisse Vor- und Nachteile besitzt.

Das in Kapitel III beschriebene Verfahren basierend auf einer linearen Regression benötigt zum Durchführen eines Fits eine Messung des Membranpotentials. Der Algorithmus lieferte in seiner ursprünglichen Form ohne Beschränkung des Parameterbereichs nur bei bestimmten Datensätze sinnvolle Ergebnisse, bei anderen war mit den ermittelten Neuronparametern keine Simulation möglich, da die Werte außerhalb des biologisch sinnvollen Bereichs lagen. Durch gezielte Verbesserungen des Verfahrens war es jedoch möglich die relativen Fehler der Neuronparameter zu reduzieren, und die Anwendbarkeit auf weitere Datensätze zu erweitern. Trotz der Modifikationen ist der relative Fehler jedoch nach wie vor vom gewählten Datensatz abhängig. Dies lässt sich dadurch begründen, dass die zusätzlichen freien Parameter bei der Approximation der Exponentialfunktion mittels Bins (siehe Abbildung III-1) teilweise eine Konvergenz gegen die ideale Lösung verhindern. Dieses Problem tritt verstärkt auf, wenn das Membranpotential ein starkes Rauschen enthält. Hinzu kommt das grundsätzliche Problem, dass das Verfahren lediglich eine lokale Anpassung durchführt - eine Anpassung auf globaler Ebene findet nicht statt. Deshalb kann es vorkommen, dass trotz relativ genauer Ermittlung der Neuronparameter das simulierte Membranpotential dennoch deutlich von den Referenzdaten abweicht (siehe Abbildung III-2). Der Versuch der Verwendung einer Metrik, um den globalen Aspekt mit in die Optimierung einzubeziehen, hatte im Test nahezu keinen Einfluss auf die relativen Fehler, sorgte jedoch dafür, dass die Anzahl der Datensätze mit unvollständigen Neuronparametern auf 0 reduziert werden konnte. Bei einem Vergleich verschiedener Stromstimuli zeigte sich, dass der Strom *Einfache Stufen* im Durchschnitt zu den geringsten relativen Fehlern führte, was vermutlich auf das nicht vorhande Rauschen sowie das Abklingen des Membranpotentials zwischen verschiedenen Anregungen zurückzuführen ist. Ein großer Vorteil dieses Algorithmus im Vergleich zu generischen Optimierungsverfahren ist der deterministische Ablauf, sowie die Verwendbarkeit zur Eingrenzung des Parameterraums für nachfolgende Algorithmen, wie es z.B. auch in Pfeil (2011) zum Einsatz kam. Als eigenständiger Optimierungsalgorithmus erweist sich diese Idee aber als noch nicht geeignet.

Bei der Analyse der in Kapitel IV vorgestellten Hillclimbing- sowie Partikelschwarm-Algorithmen zeigte sich, dass das Hillclimbing-Verfahren im Mittel schlechter abschneidet. Zum Vergleich der Spikezeiten wurde hierbei die *Viktor-Purpura*-Metrik benutzt. Das Hillclimbing-Verfahren wäre theoretisch besser, wenn die verwendete Metrik lediglich wenige lokale Maxi-

ma besitzt - dies ist bei der VP-Metrik sowie den hier verwendeten Referenzdaten jedoch offensichtlich nicht der Fall. Um diese Untersuchungen vollständig abzuschließen wäre es notwendig, ähnliche Simulationen unter Verwendung anderer Metriken durchzuführen. Ein großer Vorteil des Hillclimbing-Algorithmus wäre auch hier der deterministische Ablauf. Außerdem werden die Neuronparameter ausschließlich in Richtung des Gradienten der Metrik angepasst, während bei der PSO der Parameterraum ähnlich eines random-walks erkundet wird. Im Gegensatz zur linearen Regression sind die Ergebnisse beider Spikebasierten Optimierungsverfahren nahezu unabhängig vom verwendeten Strom.

In Abschnitt IV.3 wurde außerdem ein Algorithmus zur Abschätzung der Neuronparameter entwickelt, der durch schrittweise Simulation gewisse Nachteile der Viktor-Purpura-Metrik (z.B. dass keine eindeutige Zuordnung der Spikes möglich ist) zu umgehen versucht. In dieser Arbeit kam der Algorithmus nicht eigenständig zum Einsatz, sondern wurde in Ergänzung zum PSO und Hillclimbing-Algorithmus eingesetzt, um die Neuronparameter im Vorfeld einzugrenzen. Es zeigte sich, dass mit den dadurch gewählten Startparametern signifikant bessere Ergebnisse erziehlt werden konnten, allerdings war die Verbesserung nicht für jeden Satz von Neuronparametern gleich gut. Der Grund für die Unterschiede liegt wahrscheinlich an den verwendeten Näherungen (Abschnitt VII.2), welche dazu verwendet wurden, um die Simulationsrechenzeit speziell für niedrige Feuerraten zu reduzieren. Dass diese Näherung in der Praxis tatsächlich zu Reduktion der Laufzeit führt, konnte noch nicht verifiziert werden, da die bisherige Implementation in Python ungeeignet für einen direkten Vergleich ist. Außerdem wäre es auch denkbar, den Algorithmus mit gewissen Modifikation auf der beschleunigten BrainScaleS-Hardware durchzuführen, was aufgrund des zeitlich begrenzten Umfangs der Arbeit nicht in die Praxis umgesetzt werden konnte. Gegenüber einer einfachen Umrechnung der Neuronparameter hat das Verfahren analog zur VP-Metrik den Vorteil, dass auch dann eine Anpassung möglich ist, wenn die Hardware Abweichungen zum AdEx-Modell aufweist.

In den aktuellen Varianten existieren für alle vorgestellten Algorithmen noch gewisse Einschränkungen, die eine Anwendung auf beliebige Daten erschweren. In Bezug auf die lineare Regression wären vor allem Verbesserungen denkbar, welche mittels zusätzlicher Bedingungen an die Bins (siehe Unterabschnitt III.1.3) eine Konvergenz gegen unerwünschte Lösungen verhindern. Alternativ wäre es auch denkbar, mit mehreren Iterationen zu arbeiten, wobei jeweils die Ergebnisse der vorherigen Iteration wieder in das nächste Gleichungssystem mit einfließen, um die Genauigkeit der Approximation zu erhöhen (Vgl. *Branch-and-Cut* beim Lösen ganzzahliger linearer Optimierungsprobleme). Wie wir jedoch bereits anhand der Testszenarien gesehen haben, beträgt die Laufzeit schon bei 50 Bins bis zu mehrere Stunden - durch Hinzufügen von mehreren Iterationen bzw. Bedingungen würde sich die Laufzeit noch weiter erhöhen. Bei dem Algorithmus zur Abschätzung der Neuronparameter wäre es außerdem möglich, die vorgestellten Methoden als eigenständiges Optimierungsverfahren zu nutzen, indem man statt der Näherungslösungen eine exakte Simulation verwendet, und die Evaluationsfunktion direkt mit Partikelschwarm-Optimierung bzw. Hillclimbing minimiert. Da wir durch die Verwendung der Evaluationsfunktion einen besseren Potentialverlauf erwarten als bei Spike-Metriken, wäre dieses Verfahren eventuell besser dazu geeignet, die einem Datensatz zugrundeliegenden Neuronparameter zu finden.

Insgesamt stellte sich bei den im Rahmen dieser Arbeit analysierten Algorithmen die Kombination von Partikelschwarm-Optimierung mit dem Verfahren zur Abschätzung der Neuron-

parameter als bestes Verfahren bezüglich der Übereinstimmung mit den Referenzparametern heraus. Bei einigen Datensätzen konnte hiermit ein relativer Fehler der Parameter von unter 1 % sowie eine Viktor-Purpura-Metrik von gerundet 1.00 erreicht werden. Aus diesem Grund scheint der ursprünglich zur Abschätzung der Neuronparameter konzipierte Algorithmus auch am erfolgversprechendsten für weiterführende Untersuchungen.

VII. Anhang

VII.1. Beschränkung der Neuronparameter

Tabelle VII-1 fasst die verwendeten Beschränkungen der Neuronparameter zusammen. Die angegebenen Grenzwerte (min, max) kamen zum Einen für den linearen Regressions-Algorithmus zum Einsatz, zum Anderen wurden diese auch beim PSO-Verfahren dazu verwendet, um eine Divergenz der Parikel zu verhindern. Die Normalverteilung (mean, std) sowie das angegebene Gitter (min, max, div) wurden bei der Untersuchung spikebasierter Algorithmen zur Wahl der Startparameter verwendet. Die Spalte *div* bezeichnet dabei die Anzahl der Unterteilungen des Intervalls. Aus Zeitgründen wurden Parameter, welche für alle Referenzdaten ungefähr identisch sind, nicht genau abgetastet.

VII.2. Approximation des AdEx-Neuronmodells

Bei der Implementierung des in Abschnitt IV.3 beschrieben Verfahrens zur Abschätzung der Neuronparameter wurde für die schrittweise Simulation kein gewöhnliches Lösungsverfahren verwendet, sondern eine Approximation des AdEx-Modells, bei der der Exponentialterm linear angenähert wird (siehe Gleichung II-1). Dies hat den Grund, dass bei unserem Algorithmus weniger Wert auf Genauigkeit der Simulation gelegt wird, sondern einzig die Geschwindigkeit wichtig ist, um in kurzer Zeit die gewünschte Abschätzung zu erhalten.

$$g_{\mathrm{L}}\Delta_{\mathrm{th}} \exp\left(\frac{V - V_{\mathrm{th}}}{\Delta_{\mathrm{th}}}\right) \approx m + n \cdot V \tag{VII-1}$$

Anschließend lassen sich die Differentialgleichungen als Matrixgleichung schreiben:

$$\begin{pmatrix} \frac{\mathrm{d}V(t)}{\mathrm{d}t} \\ \frac{\mathrm{d}w(t)}{\mathrm{d}t} \end{pmatrix} = \underbrace{\begin{pmatrix} -\frac{1}{\tau_{\mathrm{m}}} + n & -\frac{1}{C} \\ \frac{a}{\tau_{\mathrm{w}}} & -\frac{1}{\tau_{\mathrm{w}}} \end{pmatrix}}_{A} \cdot \begin{pmatrix} V(t) \\ w(t) \end{pmatrix} + \begin{pmatrix} g_{\mathrm{L}} \cdot \frac{E_{\mathrm{L}}}{C} + \frac{I(t)}{C} + m \\ -a \cdot \frac{E_{\mathrm{L}}}{\tau_{\mathrm{w}}} \end{pmatrix} \tag{VII-2}$$

Unter der Annahme, dass die Matrix A zwei verschiedene Eigenwerte λ_1, λ_2 besitzt, existieren auch zwei verschiedene Eigenräume, und die Matrix ist diagonalisierbar[1]. Demnach gilt $A = S \cdot D \cdot S^{-1}$, wobei $D = diag(\lambda_1, \lambda_2)$ die Matrix der Eigenwerte ist, und S die Eigenvektoren als Spalten beinhaltet. Durch Multiplikation der obigen Gleichung mit S^{-1} von links erhält man:

[1] Dies ist praktisch für nahezu alle Matrizen erfüllt - falls nicht, kann der Term n der Approximation passend gewählt werden, sodass diese Bedingung erfüllt ist.

Tabelle VII-1.: Zusammenfassung der in Kapitel III und Kapitel IV verwendeten Beschränkungen der Neuronparameter sowie Startparameter-Verteilungen

Variable		**Grenzwerte**		**Normalverteilung**		**Gitter**		
		min	max	mean	std	min	max	div
C	[nF]	0.01	1.0	0.275	0.025	0.25	0.30	10
τ_{m}	[ms]	0.333	60.0	9.0	4.0	5	13.0	5
E_{L}	[mV]	-100.0	-40.0	-70.6	2.0	-70.6		
V_{th}	[mV]	-70.0	-10.0	-50.4	2.0	-50.4		
Δ_{th}	[mV]	0.1	20.0	2.0	0.5	2.0		
τ_{w}	[ms]	1.0	1000.0	200.0	50.0	50.0	200.0	4
a	[nS]	0.0	100.0	4.0	1.0	2.0	4.0	2
b	[nA]	0.0	2.0	0.5	0.1	0.035		
t_{ref}	[ms]	0.0	10.0	0.0	0.0	0.0		
V_{peak}	[mV]	-10.0	10.0	0.0	0.0	0.0		
V_{reset}	[mV]	-100.0	-45.0	-60.0	2.0	-60.0		
E_{k}	[mV]	-10.0	10.0	0.0	5.0	0.0		
τ_{k}	[ms]	0.0	100.0	5.0	2.0	5.0		
μ_{k}	[nS]	0.001	1.0	0.2	0.1	0.2		

$$S^{-1}\cdot\begin{pmatrix}\frac{\mathrm{d}V(t)}{\mathrm{d}t}\\ \frac{\mathrm{d}w(t)}{\mathrm{d}t}\end{pmatrix}=\underbrace{\begin{pmatrix}\lambda_1 & 0\\ 0 & \lambda_2\end{pmatrix}}_{D}\cdot S^{-1}\cdot\begin{pmatrix}V(t)\\ w(t)\end{pmatrix}+S^{-1}\cdot\begin{pmatrix}g_{\mathrm{L}}\cdot\frac{E_{\mathrm{L}}}{C}+\frac{I(t)}{C}+m\\ -a\cdot\frac{E_{\mathrm{L}}}{\tau_{\mathrm{w}}}\end{pmatrix} \tag{VII-3}$$

Unter der Annahme eines konstanten Stromes I und mithilfe der Notation

$$\begin{pmatrix}f_1(t)\\ f_2(t)\end{pmatrix}:=S^{-1}\cdot\begin{pmatrix}V(t)\\ w(t)\end{pmatrix},\begin{pmatrix}c_1\\ c_2\end{pmatrix}:=S^{-1}\cdot\begin{pmatrix}g_{\mathrm{L}}\cdot\frac{E_{\mathrm{L}}}{C}+\frac{I}{C}+m\\ -a\cdot\frac{E_{\mathrm{L}}}{\tau_{\mathrm{w}}}\end{pmatrix} \tag{VII-4}$$

erhält man die vereinfachte Gleichung:

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix}f_1(t)\\ f_2(t)\end{pmatrix}=\begin{pmatrix}\lambda_1 & \\ & \lambda_2\end{pmatrix}\cdot\begin{pmatrix}f_1(t)\\ f_2(t)\end{pmatrix}+\begin{pmatrix}c_1\\ c_2\end{pmatrix} \tag{VII-5}$$

Diese Gleichung besitzt die allgemeine Lösung:

$$f_i(t) = \left(f_i(0) + \frac{c_i}{\lambda_i} \right) \cdot \exp(\lambda_i t) - \frac{c_i}{\lambda_i} \tag{VII-6}$$

für $i \in \{1, 2\}$. Basierend auf dieser Vorgehensweise lässt sich nun der Zustand $(V(t), w(t))^{\mathrm{T}}$ zu jedem Zeitpunkt t analytisch berechnen (vorausgesetzt die obige Gleichung bleibt gültig, dazu später mehr), indem man zunächst die obige Formel verwendet, um $(f_1(t), f_2(t))^{\mathrm{T}}$ zu berechnen und das Ergebnis anschließend mit

$$\begin{pmatrix} V(t) \\ w(t) \end{pmatrix} = S \cdot \begin{pmatrix} f_1(t) \\ f_2(t) \end{pmatrix} \tag{VII-7}$$

in den Raum der Neuronzustände zurückrechnet.

Bisher wurde bei dieser Theorie jedoch noch nicht berücksichtigt, dass einerseits die zuvor angenommene Näherung des Exponentialterms nur in einem gewissen Intervall gültig ist, und zum Anderen, dass Spikes eine Singularität darstellen, und somit ebenfalls separat berücksichtigt werden müssen. Zur Lösung dieses Problems wurden die Matrizen S, D, S^{-1} mit einem Index i in Abhängigkeit des Membranpotentials V versehen (ähnlich der Bins zur Approximation in Kapitel III), sodass diese nur innerhalb eines Intervall des Membranpotentials gültig sind. Um die Erkennung von Spikes zu realisieren, und beim Verlassen des Gültigkeitsbereichs der Approximation die verwendeten Matrizen auszutauschen, wurde eine Funktion implementiert, der jeweils ermittelt, bis zu welcher Zeit t das Membranpotential innerhalb gewisser Schranken bleibt. Da keine analytische Lösung zur Bestimmenung dieser Zeit existiert, kommt in der Implementierung ein Intervallhalbierungsverfahren zum Einsatz. Um das Näherungsverfahren auch im Falle nicht-konstanter Ströme anwenden zu können, wird der Strom außerdem immer innerhalb gewisser Intervalle gemittelt (hier: jeweils im Bereich zwischen zwei aufeinanderfolgenden Spikes). Diese Vorgehensweise wird noch einmal von der nachfolgenden Abbildung VII-1 veranschaulicht. Der oben beschriebene Algorithmus basiert auf der Annahme eines Stromstimulus - mit geringfügigen Modifikationen ist es jedoch auch möglich Spikestimuli zu berücksichtigen, wenn diese Abschnittsweise in den dazugehörigen (mittleren) Strom umgerechnet werden.

VII.2.1. Vergleich mit dem Euler-Verfahren

Um den zuvor beschriebenen Algorithmus mit dem Eulerverfahren (siehe z.B. Press et al. (1992)) zu vergleichen, müssen wir zunächst einige Annahmen über den verwendeten Strom machen. Im Folgenden sei n die Anzahl der Spikes und $\overline{\Delta t}$ die durchschnittliche Zeit zwischen zwei aufeinanderfolgenden Spikes. Außerdem sei der Strom zwischen zwei aufeinanderfolgenden Spikes näherungsweise konstant, sodass eine Approximation mit dem Mittelwert möglich ist. Die Schrittweite des Euler-Verfahrens sei h. Bei dem oben vorgestellten Näherungsverfahren wählen wir als Genauigkeit der zu bestimmenden Zeiten (Austausch der Matrizen bzw. Spikes) ebenfalls h, und nehmen an, dass der Exponentitalterm mit k Matrizen approximiert wird. Da sich der Strom zwischen zwei Spikes nicht ändert steigt das Membranpotential stetig an

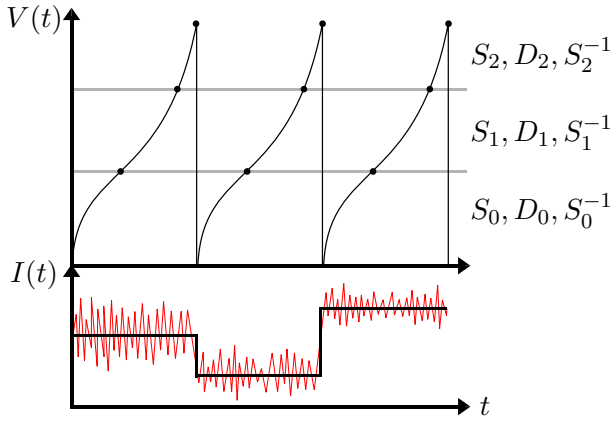


Abbildung VII-1.: Die Abbildung veranschaulicht die prinzipielle Vorgehensweise des in Abschnitt VII.2 beschriebenen Näherungsverfahrens. Statt der Lösung der exakten Differentialgleichung wird das Membranpotential V in verschiedene Intervalle unterteilt, in denen spezielle Matrizen zur Approximation gültig sind (hier 3 Matrizen). Innerhalb eines Bins ist durch die vereinfachte Annahme eines konstanten Stromstimulus statt des tatsächlichen (——) eine analytische Lösung der Differentialgleichungen möglich. Beim Verlassen der jeweiligen Gültigkeitsbereiche der Matrizen (•) müssen die Matrizen ausgetauscht werden.

und durchläuft maximal alle k Bins. Für die Komplexität beider Algorithmen erhält man also (sofern die Matrizen S_i, D_i, S_i^{-1} bereits zuvor berechnet wurden, diese Berechnung hat jedoch eine konstante Laufzeit bei fester Anzahl an Bins):

$$\begin{array}{ll} \text{Euler-Verfahren} & \mathcal{O}\left(n \cdot \frac{\overline{\Delta t}}{h}\right) \\ \text{Dieses Verfahren} & \mathcal{O}\left(n \cdot k \cdot \left(1 + \log_2\left(\frac{\overline{\Delta t}}{h}\right)\right)\right) \end{array} \tag{VII-8}$$

Die Formel für das Euler-Verfahren erhält man direkt aus der Überlegung, wie viele Zeitschritte h notwendig sind, um die gesamte Simulationsdauer $n \cdot \overline{\Delta t}$ abzudecken. Unter Annahme der obigen Voraussetzungen muss die Matrix insgesamt $n \cdot k$-mal ausgetauscht werden. Dazu ist es jeweils notwendig den genauen Zeitpunkt mittels eines Intervallhalbierungsverfahrens festzustellen. Da die Lösung mit der Genauigkeit h bestimmt werden soll, sind dazu maximal $\log_2\left(\frac{\overline{\Delta t}}{h}\right)$ Rechenschritte notwendig. Diese Überlegung zeigt, dass die Approximation für eine ausreichend niedrige Feuerrate schneller ist als das Euler-Verfahren.

In dem in Abschnitt IV.3 beschriebenen Algorithmus kam eine Näherung mit $h = 0.1$ ms sowie einer Bin-Anzahl von $k = 10$ zum Einsatz. Zur Diagonalisierung der Matrizen wurden die *NumPy*-Routinen (Oliphant, 2006) verwendet.

Literaturverzeichnis

Abbott, L. F. (1999). Lapicque's introduction of the integrate-and-fire model neuron (1907). *Brain Research Bulletin 50*(5-6), 303–304.

Born, J. (2012). Distanzmaße für Spiketrains im Rahmen der Optimierung von Neuronparametern. Bachelor thesis (German), University of Heidelberg, HD-KIP 12-18.

BrainScaleS (2011). Brain-inspired multiscale computation in neuromorphic hybrid systems. Website http://brainscales.kip.uni-heidelberg.de/.

Brette, R., & Gerstner, W. (2005). Adaptive exponential integrate-and-fire model as an effective description of neuronal activity. *J. Neurophysiol. 94*, 3637–3642.

Brownlee, J. (2011). *Clever Algorithms: Nature-Inspired Programming Recipes*. Lulu.com.

Davison, A. P., Brüderle, D., Eppler, J., Kremkow, J., Muller, E., Pecevski, D., Perrinet, L., & Yger, P. (2008). PyNN: a common interface for neuronal network simulators. *Front. Neuroinform. 2*(11).

Dayan, P., & Abbott, L. F. (2001). *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. Cambride, Massachusetts: The MIT press.

Eberhart, R., & Kennedy, J. (1995). A new optimizer using particle swarm theory. pp. 39–43.

Fransén, E., & Lansner, A. (1998). A model of cortical associative memory based on a horizontal network of connected columns. *Network — Computation in Neural Systems 9*, 235–264.

Gerstner, W., & Naud, R. (2009). How Good Are Neuron Models? *Science 326*(5951), 379–380.

Gewaltig, M.-O., & Diesmann, M. (2007). NEST (NEural Simulation Tool). *Scholarpedia 2*(4), 1430.

Hodgkin, A. L., & Huxley, A. F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *J Physiol 117*(4), 500–544.

Jolivet, R., Kobayashi, R., Rauch, A., Naud, R., Shinomoto, S., & Gerstner, W. (2008a). A benchmark test for a quantitative assessment of simple neuron models. *Journal of Neuroscience Methods 169*(2), 417 – 424.

Jolivet, R., Schürmann, F., Berger, T. K., Naud, R., Gerstner, W., & Roth, A. (2008b). The quantitative single-neuron modeling competition. *Biological Cybernetics 99*(4-5), 417–426.

Jones, E., Oliphant, T., Peterson, P., et al. (2001–). SciPy: Open source scientific tools for Python.

Kobayashi, R., Tsubo, Y., & Shinomoto, S. (2009). Made-to-order spiking neuron model equipped with a multi-timescale adaptive threshold. *Frontiers in Computational Neuroscience 3*(0).

Lundqvist, M., Rehn, M., Djurfeldt, M., & Lansner, A. (2006). Attractor dynamics in a modular network of neocortex. *Network:Computation in Neural Systems 17:3*, 253–276.

Markram, H. (2006). The Blue Brain Project. *Nature Reviews Neuroscience 7*(2), 153–160.

Mensi, S., Naud, R., Pozzorini, C., Avermann, M., Petersen, C. C. H., & Gerstner, W. (2011). Parameter extraction and classification of three cortical neuron types reveals two distinct adaptation mechanisms. *Journal of Neurophysiology*.

Naud, R., Marcille, N., Clopath, C., & Gerstner, W. (2008). Firing patterns in the adaptive exponential integrate-and-fire model. *Biological Cybernetics 99*(4), 335–347.

Nelder, J. A., & Mead, R. (1965). A simplex method for function minimization. *The Computer Journal 7*(4), 308–313.

Oliphant, T. E. (2006). *Guide to NumPy*. Provo, UT.

Paninski, L., Pillow, J. W., & Simoncelli, E. P. (2004). Maximum likelihood estimation of a stochastic integrate-and-fire neural encoding model. *Neural Computation 16*(12), 2533–2561.

Pfeil, T. (2011). Configuration strategies for neurons and synaptic learning in large-scale neuromorphic hardware systems. Diploma thesis (English), University of Heidelberg, HD-KIP 11-34.

Press, W., Teukolsky, S., Vetterling, W., & Flannery, B. (1992). *Numerical Recipes in C* (2nd ed.). Cambridge, UK: Cambridge University Press.

Rossant, C., Goodman, D. F., Platkiewicz, J., & Brette, R. (2010). Automatic fitting of spiking neuron models to electrophysiological recordings. *Frontiers in Neuroinformatics 4*.

Stark, P. B., & Parker, R. L. (1995). Bounded-variable least-squares: an algorithm and applications. *Computational Statistics*.

Victor, J. D., & Purpura, K. P. (1996). Nature and precision of temporal coding in visual cortex: a metric-space analysis. *J Neurophysiol 76*(2), 1310–1326.

Weise, T. (2009). *Global Optimization Algorithms - Theory and Application* (Second ed.). Self-Published. Online available at http://www.it-weise.de/.

Danksagungen

Abschließend möchte ich mich noch bei allen Personen bedanken, die mich während der Entstehung dieser Bachelorarbeit unterstützt haben.

Ein großer Dank geht an:

Herrn Prof. Dr. Meier	für die freundliche Aufnahme in die Arbeitsgruppe sowie die Unterstützung während meiner Bachelorarbeit
Herrn Dr. Schemmel	dafür mein Zweitprüfer zu sein
Thomas Pfeil	für die vorbildliche Betreuung, die durchweg nützlichen Tipps, sowie die unendliche Geduld beim Probelesen dieser Arbeit
Christoph Koke & Sebastian Jeltsch	ebenfalls für Ihre Unterstützung und Ausdauer im Probelesen
Ioannis Kokkinos & Anne-Christine Scherzer	für die netten Gespräche und gute Ratschläge
Eric Müller	für die Hilfe und Beratung bei technischen Problemen aller Art
alle anderen der Visions-Gruppe	für die freundliche Atmosphäre, wodurch das Arbeiten in der Gruppe durchweg Spaß gemacht hat
Michael Müller	für das Probelesen dieser Arbeit und die tolle moralische Unterstützung
meine Eltern	die mich während des gesamten Studiums unterstützt haben, und ohne die diese Arbeit nie möglich gewesen wäre

alle weiteren Helfer, die mich unterstützt haben, aber in der Eile leider vergessen wurden.

Erklärung

Ich versichere, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Heidelberg, den 6. August 2012

. . . . . .

(Unterschrift)